



***Microtec  
Research Inc.***

# ***ASM68K***

## ***User's Guide***

100010-007

## TRADEMARKS

Apollo™, DOMAIN/OS™, and AEGIS™ are trademarks of Apollo Computer, Inc.

Flexible License Manager™ is a trademark of Highland Software, Inc.

High C® is a registered trademark of Metaware, Inc.

HP™ and HP-UX™ are trademarks of Hewlett-Packard Company.

IBM® is a registered trademark of International Business Machines, Inc.

ICE® is a registered trademark of Intel Corporation.

Microtec® and Paragon® are registered trademarks of Microtec Research, Inc.

MRI ASM™, MRI C™, MRI FORTRAN™, MRI Pascal™, and MRI XRAY™ are trademarks of Microtec Research, Inc.

MS-DOS™ is a trademark of Microsoft Corporation.

pSOS™ is a trademark of Software Components Group.

Sun™ is a trademark of Sun Microsystems, Inc.

Tektronix® is a registered trademark of Tektronix, Inc.

UNIX® is a registered trademark of AT&T.

VAX™, VMS™, and ULTRIX™ are trademarks of Digital Equipment Corporation.

VRTX™ is a trademark of Ready Systems, Inc.

XRAY68K™, XRAY86™, XRAY180™, XRAYZ80™, XRAY29K™, XRAY51™, XRAY88K™, XRAY960™, XRAYG32™, XRAYTX™, XRAY Debugger™, XRAY In-Circuit Debugger™, and XRAY In-Circuit Debugger Monitor™, XHM302™, XRAYH83™, and XRAYH85™ are trademarks of Microtec Research, Inc.

Zilog® is a registered trademark of Zilog Corporation.

302bug™ is a trademark of Motorola, Inc.

## RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013 of DOD Regulation 27-401, or comparable clause, as such may be amended or renumbered from time to time.

MICROTEC RESEARCH, INCORPORATED  
2350 MISSION COLLEGE BLVD.  
SANTA CLARA, CA 95054

© Copyright 1987, 1988, 1989, 1990, 1991 Microtec Research, Inc. All rights reserved. No part of this publication may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical or otherwise, without prior written permission of Microtec Research, Inc.

| REV. | REVISION HISTORY                                                                                                                           | DATE  | APPD. |
|------|--------------------------------------------------------------------------------------------------------------------------------------------|-------|-------|
| -002 | Updated from V 6.3 to V 6.4 the VAX/VMS and UNIX chapters.                                                                                 | 6/88  | L.C.  |
| -003 | Updated from V 6.4 to V 6.5 the VAX/VMS and UNIX chapters.<br><br>Added the IBM-PC/MS-DOS chapter.                                         | 8/89  | L.C.  |
| -004 | Printed at 80% for new packaging.                                                                                                          | 11/89 | L.C.  |
| -005 | For V 6.4, merged the IBM-PC/MS-DOS chapter into this guide.<br><br>Printed V 6.4 at 80% for new packaging.                                | 12/89 | L.C.  |
| -006 | Updated from V 6.5 to V 6.6 the VAX/VMS, UNIX, and IBM-PC/MS-DOS chapters.<br><br>Added the LLEN and P=68332 assembler command line flags. | 2/90  | L.C.  |
| -007 | Updated to Version 6.8A.                                                                                                                   | 9/91  | L.L.  |



## About the Documentation Set

This manual is part of a document set that describes the operation of the software provided with the Microtec Research assembler package. The *ASM68K Documentation Set* includes the following:

1. The *User's Guide* provides host computer and operating system-specific descriptions of how to invoke the assembler, linker, and object module librarian.
2. The *Reference Manual* contains complete descriptions of the assembler syntax and directives, linker commands, and librarian commands. Sample program listings are used as examples for the assembler, linker, and object module librarian. Error and warning messages are described.
3. The *Installation Guide* contains host computer and operating system-specific directions for installing the ASM68K package.

Once the programs have been installed, the *User's Guide* can be used in conjunction with the *Reference Manual*.

## About this Guide

The Microtec Research *ASM68K User's Guide* describes how to use Microtec Research's ASM68K assembler package software. Detailed descriptions are given of the various ways to invoke the assembler, linker, and object module librarian. Additionally, command line options that let you control output listing and object module generation for the programs are described.

This guide contains the following chapters and appendices:

- Chapter 1, *Introduction*, provides a high-level description of the assembler, linker, object module librarian, the optional XRAY68K debugger package, and the optional MCC68K Compiler.
- Chapter 2, *VMS User's Guide*, describes how to invoke the assembler, linker, and object module librarian. Command syntax and options are provided for a VAX host computer using the VMS operating system.
- Chapter 3, *UNIX/DOS User's Guide*, describes how to invoke the assembler, linker, and object module librarian. Command syntax and

options are provided for host computers using a UNIX or UNIX-like operating system, PC-DOS, or MS-DOS.

- Appendix A, *HP 64000 Development System Support*, describes how to use the ASM68K assembler package to generate HP 64000 object modules that can be downloaded and used with various Hewlett-Packard products.

## Notational Conventions

Examples in this guide are identified by operating system name. The name UNIX refers to any UNIX or UNIX-like operating system, such as SunOS and HP-UX. The name DOS refers to both MS-DOS and PC-DOS systems.

This guide uses the following conventions shown in Table P-1 (unless otherwise noted).

Table P-1. Notational Conventions

| Symbol                       | Name            | Usage                                                                                                            |
|------------------------------|-----------------|------------------------------------------------------------------------------------------------------------------|
| { }                          | Curly Braces    | Encloses a list from which you must choose an item.                                                              |
| [ ]                          | Square Brackets | Encloses items that are optional.                                                                                |
|                              | Vertical Bar    | Separates items in a list.                                                                                       |
| ...                          | Ellipsis        | Indicates that you may repeat the preceding item zero or more times.                                             |
| <code>typewriter font</code> |                 | Represents user input in interactive examples.                                                                   |
| <code>punctuation</code>     |                 | Punctuation other than that described here must be entered as shown.                                             |
| <i>italics</i>               |                 | Indicates a descriptive item that should be replaced with an actual item. All file name examples are in italics. |

## Microprocessor References

The 68000 family of microprocessors includes the 68000, 68008, 68010, 68012, 68020, 68030, 68302, CPU32, and 68040. In this manual, they are referred to collectively as 68000.

## Questions and Suggestions

To help us respond to questions or suggestions regarding the ASM68K assembler package, we have provided two response forms in the back of this guide.

The first form is a Reader's Response sheet, which is used to help us correct and improve our documentation. We always appreciate your comments, and by completing this form, you can participate directly in the revisions of future publications. The Reader's Response sheet is directed to Technical Publications.

The second form is a Software Performance Report which should be completed if you encounter any errors or problems with Microtec Research software. This report is directed to Technical Support.





# Contents

---

## Preface

|                                   |     |
|-----------------------------------|-----|
| About the Documentation Set ..... | v   |
| About this Guide .....            | v   |
| Notational Conventions .....      | vi  |
| Microprocessor References .....   | vi  |
| Questions and Suggestions .....   | vii |

## 1 Introduction

|                                                          |     |
|----------------------------------------------------------|-----|
| Components of the Microtec Research ASM68K Package ..... | 1-1 |
| Assembler .....                                          | 1-1 |
| Linker .....                                             | 1-1 |
| Object Module Librarian .....                            | 1-2 |
| Other Microtec Research Products .....                   | 1-2 |
| MCC68K C Compiler .....                                  | 1-2 |
| CCC68K C++ Compiler .....                                | 1-2 |
| XRAY68K Debugger .....                                   | 1-3 |
| Data Flow .....                                          | 1-3 |
| Software Development Cycle .....                         | 1-5 |
| Native Environment .....                                 | 1-6 |
| Cross Environment .....                                  | 1-7 |
| Embedded Environment .....                               | 1-9 |

## 2 VAX / VMS User's Guide

|                                              |      |
|----------------------------------------------|------|
| Introduction .....                           | 2-1  |
| ASM68K Assembler .....                       | 2-1  |
| LNK68K Linker .....                          | 2-1  |
| LIB68K Librarian .....                       | 2-2  |
| ASM68K Assembler .....                       | 2-2  |
| Invocation Syntax .....                      | 2-2  |
| File Name Defaults .....                     | 2-4  |
| Command Line Flags .....                     | 2-5  |
| Invocation Examples .....                    | 2-11 |
| LNK68K Linker .....                          | 2-12 |
| Invocation Syntax .....                      | 2-12 |
| File Name Defaults .....                     | 2-15 |
| Invocation Examples .....                    | 2-16 |
| Invoking LNK68K with a Command File .....    | 2-16 |
| Invoking LNK68K Without a Command File ..... | 2-17 |
| LIB68K Librarian .....                       | 2-18 |
| Invocation Syntax .....                      | 2-18 |

|                                             |      |
|---------------------------------------------|------|
| File Name Defaults .....                    | 2-19 |
| Invocation Examples .....                   | 2-20 |
| Invoking LIB68K Interactively .....         | 2-20 |
| Invoking LIB68K from the Command Line ..... | 2-20 |
| Invoking LIB68K with a Command File .....   | 2-21 |
| VMS Return Codes .....                      | 2-21 |

### 3 UNIX / DOS User's Guide

|                                              |      |
|----------------------------------------------|------|
| Introduction .....                           | 3-1  |
| ASM68K Assembler .....                       | 3-1  |
| LNK68K Linker .....                          | 3-1  |
| LIB68K Librarian .....                       | 3-2  |
| ASM68K Assembler .....                       | 3-2  |
| Environment Variables .....                  | 3-2  |
| TMP (DOS only) .....                         | 3-2  |
| Invocation Syntax .....                      | 3-2  |
| File Name Defaults .....                     | 3-4  |
| Command Line Flags .....                     | 3-5  |
| Invocation Examples .....                    | 3-12 |
| LNK68K Linker .....                          | 3-13 |
| Environment Variables .....                  | 3-13 |
| MRI_68K_LIB .....                            | 3-13 |
| TMP (DOS only) .....                         | 3-14 |
| Invocation Syntax .....                      | 3-14 |
| File Name Defaults .....                     | 3-17 |
| Invocation Examples .....                    | 3-17 |
| Invoking LNK68K with a Command File .....    | 3-17 |
| Invoking LNK68K Without a Command File ..... | 3-19 |
| LIB68K Librarian .....                       | 3-20 |
| Invocation Syntax .....                      | 3-20 |
| File Name Defaults .....                     | 3-21 |
| Invocation Examples .....                    | 3-22 |
| Invoking LIB68K Interactively .....          | 3-22 |
| Invoking LIB68K from the Command Line .....  | 3-22 |
| Invoking LIB68K with a Command File .....    | 3-23 |
| UNIX/DOS Return Codes .....                  | 3-23 |
| Conversion Utility .....                     | 3-24 |

### Appendix A: HP 64000 Development System Support

|                               |     |
|-------------------------------|-----|
| Overview .....                | A-1 |
| HP 64000 Considerations ..... | A-1 |

## Figures

---

|             |                                                                         |     |
|-------------|-------------------------------------------------------------------------|-----|
| Figure 1-1. | Components of the Microtec Research ASM68K Package .....                | 1-4 |
| Figure 1-2. | Products Used in Coding and Testing Phases (Native Environment) .....   | 1-6 |
| Figure 1-3. | Products Used in Coding and Testing Phases (Cross Environment) .....    | 1-8 |
| Figure 1-4. | Products Used in Coding and Testing Phases (Embedded Development) ..... | 1-9 |

## Tables

---

|            |                                                       |      |
|------------|-------------------------------------------------------|------|
| Table P-1. | Notational Conventions .....                          | vi   |
| Table 2-1. | VMS Default Assembler File Name Extensions .....      | 2-4  |
| Table 2-2. | VMS Assembler Command Line Flags .....                | 2-5  |
| Table 2-3. | VMS Default Linker File Name Extensions .....         | 2-15 |
| Table 2-4. | VMS Default Librarian File Name Extensions .....      | 2-19 |
| Table 3-1. | UNIX/DOS Default Assembler File Name Extensions ..... | 3-4  |
| Table 3-2. | UNIX/DOS Assembler Command Line Flags .....           | 3-5  |
| Table 3-3. | UNIX/DOS Default Linker File Name Extensions .....    | 3-17 |
| Table 3-4. | UNIX/DOS Default Librarian File Name Extensions ..... | 3-21 |
| Table A-1. | Section Types for the HP 64000 Sections .....         | A-2  |
| Table A-2. | Modifications Required for the 68000 Monitor .....    | A-3  |



# Introduction 1

---

## Components of the Microtec Research ASM68K Package

The Microtec Research ASM68K Assembler package consists of an assembler, a linker, and an object module librarian. These components are described in the following sections.

### Assembler

The Microtec Research ASM68K Assembler converts assembly language programs to relocatable object code. Object modules are suitable for linking with other modules or with libraries of modules.

The assembler accepts source program statements that are syntactically compatible with those accepted by Motorola 68000 family assemblers. Many additional directives are provided for versatility. The ASM68K Assembler processes macros, conditional assembly statements, and compiler-like structure directives. The assembler generates a cross-reference table as well as a standard symbol table. Generated code and data can be placed in multiple named or numbered sections.

After assembly, the linker links the object modules, which are then suitable for download and execution on any Motorola 68000 family microprocessor.

### Linker

The Microtec Research LNK68K Linker combines relocatable object modules into a single absolute object module. You can optionally generate object modules in the following formats:

- Motorola S-Record
- Hewlett-Packard 64000 format (HP-OMF)
- IEEE-695

The linker also supports the combining of multiple relocatable object modules into a single relocatable module, which subsequently can be re-linked with other modules. This feature is referred to as incremental linking.

If one of the input files is a library of object modules, the linker automatically loads only those modules from the library that are referenced by the other named object modules. The linker produces a link map that shows the final location of all modules and sections, and the final absolute values of all symbols. A cross-reference listing shows which modules refer to each global symbol.

You can execute the linker in batch mode by using a command file or in a modified batch mode by specifying a command file plus additional object modules/libraries on the command line. The linker reports unresolved external symbols as well as errors that occur at link time, and prints these messages on the link map or on the terminal.

## Object Module Librarian

The Microtec Research LIB68K Object Module Librarian lets you maintain a collection of relocatable object modules that reside in one file. Libraries let you automatically load frequently-used object modules without concern for the specific names and characteristics of the modules.

By using the librarian, you can format and organize library files that will subsequently be used by the linker. You can add, delete, replace and extract modules, as well as obtain a directory listing of library contents.

## Other Microtec Research Products

### MCC68K C Compiler

Microtec Research compilers emit either object code or assembly language code and optionally emit debugging information. To use the XRAY68K Debugger in high-level mode, you must compile programs with the debug option. This option causes the compiler to produce the required debugging information and pass it to the ASM68K Assembler, or the LNK68K Linker if object code is emitted directly by the compiler.

### CCC68K C++ Compiler

The Microtec Research C++ Compiler is composed of a C++ preprocessor, C++ translator, and the MCC68K ANSI C Compiler. When you invoke the CCC68K C++ Compiler, you automatically invoke the C++ preprocessor and translator which translates your C++ code into C code. The MCC68K Compiler will then compile this C code into ASM68K assembly code or object code. If you use the compiler debug option, information necessary for debugging with the XRAY68K Debugger is included in the output file.

## XRAY68K Debugger

The XRAY Debugger lets you monitor and control the execution of high-level C, Pascal, or FORTRAN programs or assembly-level programs in the controlled environment of the host computer, thereby eliminating hardware-induced anomalies. By using XRAY, you can perform testing and debugging in advance of target hardware implementation as well as during periods of hardware unavailability.

XRAY lets you examine or modify the value of program variables using the same high-level or assembly-level terms, definitions, and structures that were defined in the original source code. To display information, XRAY employs a window-oriented user interface that segregates debugging information into meaningful sections. You can even define your own windows.

The debugger is interactive and gives you complete control of the program. It executes your program while allowing you to access variables, procedures, source line addresses and other program entities.

XRAY's powerful command language allows simple and complex breakpoint setting, single-stepping, and continuous variable monitoring. By simulating memory-mapped port I/O and interrupts, XRAY lets you redirect input and output to/from files, buffers, or viewports. In addition, a sophisticated macro facility allows complex command sequences to be associated with events such as the execution of a specific statement or the accessing of a specific data location. These features let you isolate errors and patch your source code.

## Data Flow

Figure 1-1 shows the components of the Microtec Research ASM68K package and the way in which they communicate with each other and with the host computer system.

Microtec Research provides compilers, assemblers, linkers, librarians, debuggers for simulation, and debuggers with monitor capabilities for the native environment as well as the cross-environment. Products contain numerous features which are needed for developing high-performance embedded applications. These products form the 68000 toolkit.

Although embedded applications are becoming more common, many developers are not familiar with the unique problems caused by this approach. The following section describe the standard software development cycle and issues that should be considered when developing embedded applications.

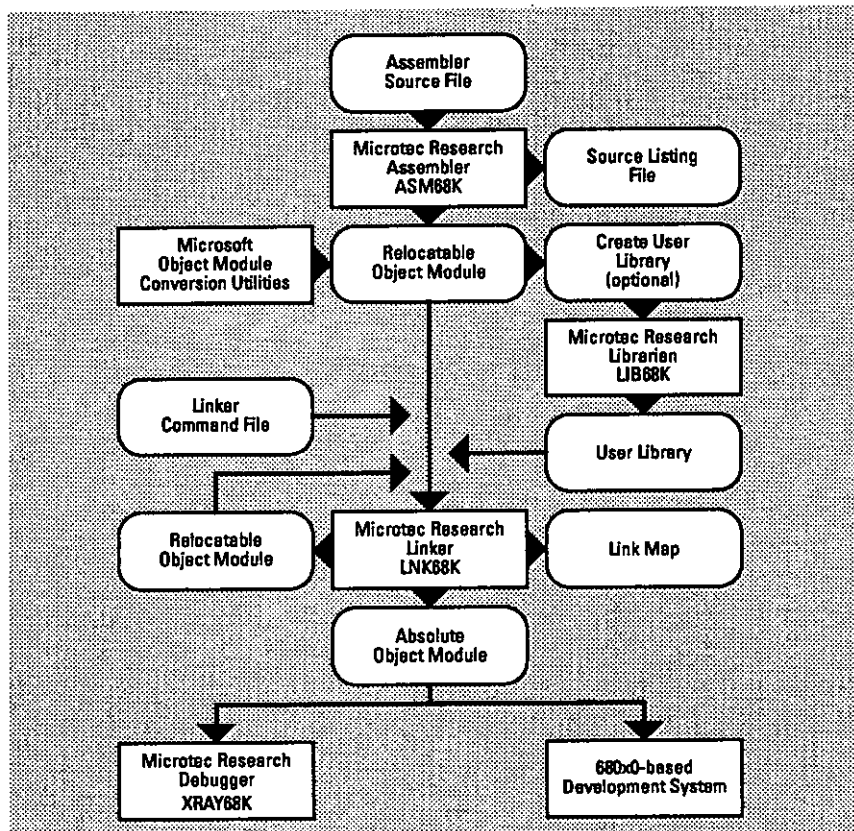


Figure 1-1. Components of the Microtec Research ASM68K Package



## Software Development Cycle

The standard software development cycle includes the following phases: requirements, design, coding, and testing. Once the product specifications have been defined and the design phase has completed, the software developer proceeds to implement this design in the coding phase and verifies that it works correctly in the testing phase. This section describes the coding and testing phases of a product and the various utilities used in these phases.

Figure 1-2 shows a high-level block diagram of the coding and testing phases. You create your software program using an editor. Library routines are put into libraries or taken out of libraries using the librarian. The program is then compiled, assembled, and linked to produce executable code. The coding phase is now complete and the testing phase begins. If any errors are found in the coding or testing phase, you return to the editor to modify your program and continue through the cycle as before.

## Native Environment

Normally, coding, testing, and the eventual execution of the program occurs on one machine. Any utility that is designed to operate on one type of machine and produce output for that type of machine is a native utility. For example, if you are on a machine and use a compiler that creates object code for that machine, that compiler is called a native compiler.

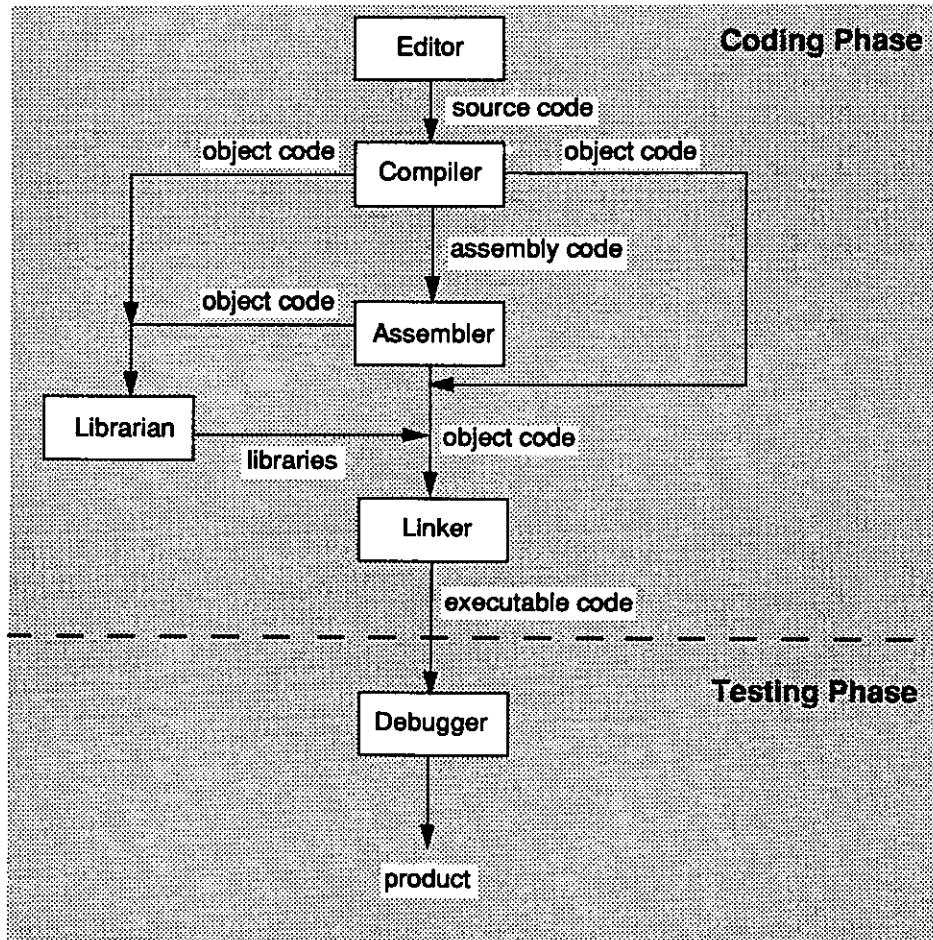


Figure 1-2. Products Used in Coding and Testing Phases  
(Native Environment)

## Cross Environment

As more computers were designed using different chips, software developers were faced with the problem of developing software on one machine and executing this software on a different machine. Utilities that operate on one machine and generate code for a different type of machine are known as cross utilities. For example, if you are on a 68000 machine and use a compiler that creates 8086 object code, that compiler is called a cross-compiler. Your 68000 machine is referred to as the "development" environment and the 8086 machine is referred to as the "target" environment.

The products used in the coding and testing phases are shown in Figure 1-3. With the exception of the editor, all the other products (compiler, assembler, linker, librarian, and debugger) generate or read target machine code instead of development machine code. For example, if your development machine is 68000-based and your target machine is 8086-based, follow the development steps below:

- Create a program in a high-level language (e.g., C) on your development machine using a standard editor.
- Generate 8086 object code or assembly code using an 8086 cross-compiler on your high-level source program.
- Assemble your 8086 assembly code using an 8086 cross-assembler.
- Combine 8086 object code library routines by using an 8086 librarian.
- Link your 8086 modules together to form an 8086 executable using an 8086 linker.
- Debug your 8086 executable using a monitor, simulator or emulator, while still residing on your 68000 development machine.
- Download your program to the target 8086 machine for more tests once you have debugged your software.

In a standard cross-environment, a target machine is similar to your development machine even though it is based on a different hardware architecture. Both machines have an operating system, terminals, disks, etc.

A need for hardware products to perform specific functions led to embedded systems. An embedded application performs specific tasks and executes on a CPU located in the middle of some other type of hardware. Since your software is not located on a normal machine, embedded software development creates some unique problems. The next section describes the embedded software development cycle and some of the unique problems that occur.

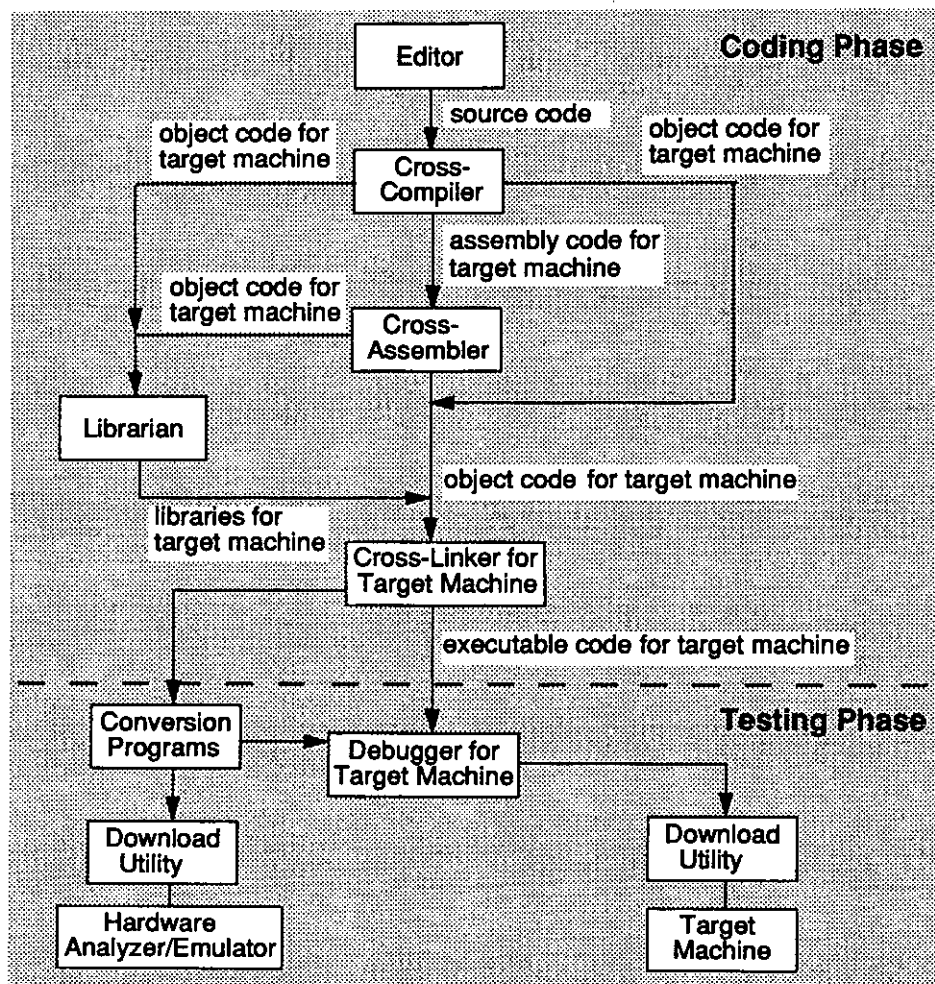


Figure 1-3. Products Used in Coding and Testing Phases  
(Cross Environment)

## Embedded Environment

Software created for an embedded application is usually located in RAM and ROM within the hardware unit that it serves. This hardware unit may or may not have an operating system, display terminal, storage devices, etc. It may interface to a variety of peripherals. Restraints on space, weight, and speed may force the developer to deal with limited memory and real-time performance requirements. Figure 1-4 illustrates the coding and testing phases for the embedded application.

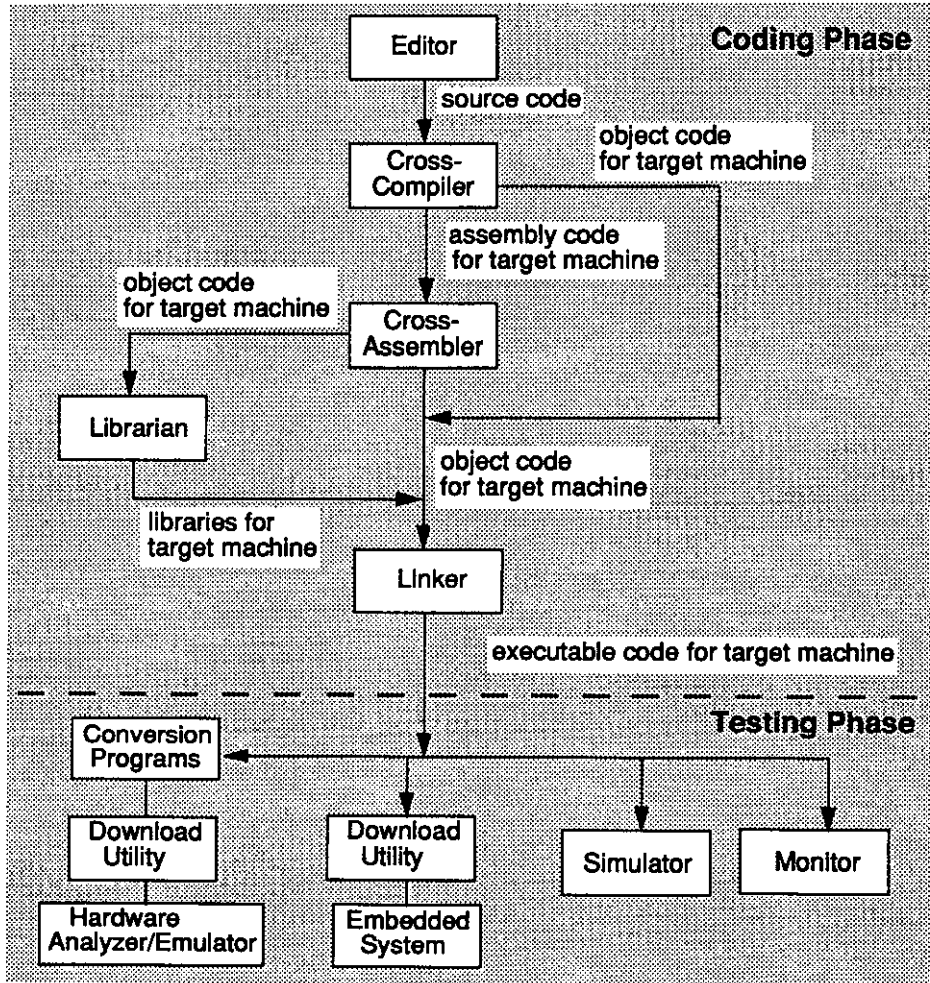


Figure 1-4. Products Used In Coding and Testing Phases (Embedded Development)

Despite similarities to cross-environment applications, this new environment, the embedded environment, forces the developer to consider problems that do not occur in cross-environments or native environments:

- no high-level operating system
- non-standard I/O
- interfaces to unique peripherals
- physical location of program in target machine
- initialization of program in target machine
- different file formats
- debugging capabilities
- downloading code to target machine's RAM or ROM

These problems, coupled with the difficulty of embedded testing, make embedded applications a challenging task. Often, the embedded hardware is not available until late in the development cycle and you are forced to develop your software by using simulators, or downloading your software to hardware test machines or emulators. Sets of tools that are designed for the embedded environment, such as the 68000 toolkit, provide the flexibility and versatility required by this unique environment.

# VAX / VMS User's Guide 2

---

## Introduction

This chapter describes the syntax for invoking the assembler, linker, and object module librarian on a VAX host computer under the VMS operating system. Command line options and their syntax are described. Command line options control the generation of the output listing and object module, and turn assembler switches on and off.

The operating system prompt used in the directions is a dollar sign (\$). Your system may be configured with a different prompt.

## ASM68K Assembler

The Microtec Research ASM68K Assembler converts assembly language programs to relocatable object code. Object modules are produced in IEEE-695 format. After assembly, object modules are combined by the linker.

## LNK68K Linker

The Microtec Research LNK68K Linker allows multiple relocatable object modules to be combined into a single relocatable module that subsequently can be re-linked with other modules (i.e., incremental linking). The linker can also combine relocatable object modules into a single absolute object module in one of the following formats:

- Motorola S-Record format
- IEEE-695 format
- Hewlett-Packard 64000 format (HP-OMF)

The LNK68K Linker has some powerful features designed for embedded applications, which provide the capability to load your software at locations that match your unique hardware environment. You can specify:

- where to load segments
- order to load segments
- size of segments to use

If one or more of the linker input files is a library of object modules, the linker automatically loads all modules from the library that are referenced by the individually-named object modules.

## LIB68K Librarian

The Microtec Research LIB68K Object Module Librarian lets you maintain a collection of relocatable object modules that reside in one file. By using the librarian, you can format and organize library files that will subsequently be used by the linker.

## ASM68K Assembler

The Microtec Research ASM68K Assembler produces object code that will be used later by the LNK68K Linker and LIB68K Librarian.

## Invocation Syntax

This section describes the command line syntax needed to invoke the ASM68K Assembler. Abbreviations for options are shown in the *Description* section. After entering an abbreviation, any number of characters can be specified up to and including the option's full spelling.

Use parentheses to specify multiple settings for the same option. For example, /IPATH=(/dir1/dir2) will search both directories, as specified. However, if you had entered /IPATH=dir1 /IPATH=dir2, only the last parameter is used; the first directory is not searched.

Options are case-insensitive and are separated by a slash (/).

### Syntax:

```
A68K [ /DEFINE sym(=value) ] [ /FLAGS="flag_list" ]
      [ /HP | /NOHP ] [ /IPATH=dir ] [ /LIST[=list_file] | /NOLIST ]
      [ /OBJECT[=object_file] | /NOOBJECT ]
      [/VERSION] source_file
```

### Description:

**A68K**                    The invocation name of the assembler.

**DEFINE *sym*(=*value*)**

Defines the symbol *sym* at assembly time. The parameter *value* must be a constant. If *value* is not specified, *sym* is set to 1.

The value of *sym* will be overridden by the first SET directive applied to *sym* in the source file. No EQU directives can be applied to *sym*.



**FLAGS="flag\_list"** Enables and disables the assembler control flags (see Table 2-2). This option may be specified before or after the *source\_file*.

**HP**  
**H**

Produces an HP 64000 compatible (HP-OMF) *asmb\_sym* symbol file. The **D** flag is automatically set, placing local symbols in the output object module. The default file name is *source\_file.A*. This option may be specified before or after the *source\_file*.

For more information on HP 64000 development support, see Appendix A.

**IPATH=dir**

Specifies a path to be searched to locate files that are to be included using the **INCLUDE** directive. For more information on the **INCLUDE** directive, see *Assembler Directives* in the *Microtec Research ASM68K Reference Manual*.

The assembler first searches for **INCLUDE** files in the directory containing the assembler source program. It next searches the directory specified by *dir*. A maximum of four paths can be specified.

**LIST[=list\_file]**

Produces a listing output file with the file name indicated. This option may be specified before or after the *source\_file*.

By default, a listing file is produced with the name *source\_file.LIS*.

**NOHP (default)**

Does not produce an HP 64000 compatible (HP-OMF) *asmb\_sym* file. This option may be specified before or after the *source\_file*.

For more information on HP 64000 development support, see Appendix A.

**NOLIST**

Inhibits production of a listing file. This switch overrides **OPT S** directives in the assembler source file. This option may be specified before or after the *source\_file*.

**NOOBJECT**

Inhibits production of an output object module. This switch overrides **OPT O** directives in the assembler source file. This option may be specified before or after the *source\_file*.

**OBJECT [=object\_file]**

Produces an output object module with the file name indicated. This option may be specified before or after the *source\_file*.

By default, an object file is produced with the name *source\_file*.OBJ.

**VERSION**

Displays the version number of ASM68K and then exits.

**source\_file**

Specifies the assembler input file. The default file name extension is .SRC.

If you do not specify options, the following defaults will be used:

```
/NOBP
/LIST=source_file.LIS
/OBJECT=source_file.OBJ
```

Refer to Table 2-2 for the default flags that will be used.

## File Name Defaults

Specification of the object file and list file are optional. If they are not specified, the assembler produces them by default. The assembler uses the source file name as the root file name and appends the default extensions listed in Table 2-1.

Table 2-1. VMS Default Assembler File Name Extensions

| File                   | Extension |
|------------------------|-----------|
| HP 64000 asmb_sym file | .A        |
| Listing file           | .LIS      |
| Object file            | .OBJ      |
| Source file            | .SRC      |

File names can include some or all of the following specifications:

- Node name (current default assumed)
- Device name (current default assumed)
- Directory (current default assumed)
- File name
- Extension
- Version number

## Command Line Flags

Flags entered on the command line enable and disable internal assembly control switches. If you use a flag, the corresponding switch is enabled. Assembler command line flags for *flag\_list* in the `FLAGS="flag_list"` option are listed in Table 2-2.

Abbreviations for the flags are shown. After entering an abbreviation, any number of characters can be specified up to the flag's full spelling. Parentheses, when shown, are required. Some flags can be negated by preceding the flag with the word **NO** or a minus sign (-). Multiple flags must be separated with commas and enclosed in double quotes. Examples of how to specify multiple flags can be found in the section titled *Assembler Invocation Examples*.

You can obtain more information about the flags in the description of the **OPT** directive in the *Microtec Research ASM68K Reference Manual*. Flags entered on the command line have the effect of inserting an **OPT** command at the beginning of the source file. However, flags on the command line do not override other **OPT** directives in the source file.

Table 2-2. VMS Assembler Command Line Flags

| Flag     | Meaning (positive form)                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ABSPCADD | When this flag is set, an absolute expression in conjunction with the mnemonic <b>PC</b> (i.e., <b>5(PC)</b> ) refers to an absolute address accessed through <b>PC</b> relative mode rather than to a relative displacement to the current program counter.<br><br>For more information, refer to <i>Assembler Syntax for Effective Address Fields</i> in Chapter 4, <i>Addressing Modes</i> , of your <i>ASM68K Reference Manual</i> .<br>(default: ABSPCADD) |
| BRB      | Forces forward references in relative branch instructions ( <b>Bcc</b> , <b>BRA</b> , <b>BSR</b> ) to use the short form of the instruction (8-bit displacement). <b>BRB</b> is legal only in 68020 mode.<br>(default: NOBRB)                                                                                                                                                                                                                                   |
| BRL      | Forces forward references in relative branch instructions ( <b>Bcc</b> , <b>BRA</b> , <b>BSR</b> ) to use the long address (32 bits). If <b>OPT OLD</b> is in effect in conjunction with the <b>BRL</b> flag, the longer form is used. This flag only applies to 68020 and 68030 processors.<br>(default: NOBRL)                                                                                                                                                |

(cont.)

Table 2-2. VMS Assembler Command Line Flags (cont.)

| Flag                     | Meaning (positive form)                                                                                                                                                                                                             |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>BRS</b><br><b>B</b>   | Forces forward references in relative branch instructions ( <i>Bcc</i> , <i>BRA</i> , <i>BSR</i> ) to use the short form of the instruction (8-bit displacement).<br>(default: <i>NOBRS</i> )                                       |
| <b>BRW</b>               | Forces 16-bit displacements to always be used in relative branch instructions ( <i>Bcc</i> , <i>BRA</i> , <i>BSR</i> ) that have forward references.<br>(default: <i>BRW</i> )                                                      |
| <b>CASE</b><br><b>CA</b> | Retains case-sensitivity of symbols. For example, <i>abc</i> and <i>ABC</i> are treated as different symbols.<br>(default: <i>CASE</i> )                                                                                            |
| <b>CEX</b><br><b>C</b>   | Lists all lines of object code that are generated by the <i>DC</i> directive. The <i>NOCEX</i> flag only lists the first line of a <i>DC</i> directive.<br>(default: <i>CEX</i> )                                                   |
| <b>CL</b><br><b>I</b>    | Lists instructions that are not assembled due to conditional assembly statements.<br>(default: <i>CL</i> )                                                                                                                          |
| <b>CRE</b>               | Lists the cross-reference table on the output listing. This option overrides the symbol table output option, <i>T</i> . If both <i>T</i> and <i>CRE</i> are used, a cross-reference table is generated.<br>(default: <i>NOCRE</i> ) |
| <b>D</b>                 | Places local symbols in the output object module. This flag is automatically set if you generate an <i>asmb_sym</i> file by using the <i>HP</i> option.<br>(default: <i>NOD</i> )                                                   |
| <b>E</b>                 | Lists lines with errors on your terminal as well as on the output listing.<br>(default: <i>E</i> )                                                                                                                                  |

(cont.)

Table 2-2. VMS Assembler Command Line Flags (cont.)

| Flag           | Meaning (positive form)                                                                                                                                                                                                                                             |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FRL            | Forces instructions that contain a forward reference to an absolute (non-relocatable) address to use a 32-bit address instead of a 16-bit address.<br>(default: 32-bit address or FRL)                                                                              |
| FRS<br>F       | Forces instructions that contain a forward reference to an absolute (non-relocatable) address to use a 16-bit address instead of a 32-bit address.<br>(default: NOFRS)                                                                                              |
| G              | Lists assembler-generated symbols in the symbol or cross-reference table. If D is also specified, these symbols are included in the object module as well.<br>(default: NOG)                                                                                        |
| LLEN= <i>n</i> | Sets the length of the line on the source listing to be <i>n</i> columns long. The <i>n</i> can be a number between 37 and 1120, inclusive.<br>(default: LLEN=132)                                                                                                  |
| MC             | Prints macro calls on the program listing.<br>(default: MC)                                                                                                                                                                                                         |
| MD             | Prints macro definitions on the program listing.<br>(default: MD)                                                                                                                                                                                                   |
| MEX<br>M       | Lists macro and structured control directive expansions on the program listing.<br>(default: MEX)                                                                                                                                                                   |
| NEST= <i>n</i> | Sets <i>n</i> nesting levels for macros. The maximum number of nesting levels allowed is 100.<br>(default: NEST=100)                                                                                                                                                |
| O              | Produces the output object module.<br>(default: O)                                                                                                                                                                                                                  |
| OLD            | Forces the assembler to generate 16-bit displacements for Bcc instructions when OPT BRL or OPT -B are specified, or when explicit .L qualifiers are used in 68020 mode. This is useful when migrating 68000 programs to the 68020 microprocessor.<br>(default: OLD) |

(cont.)

Table 2-2. VMS Assembler Command Line Flags (cont.)

| Flag                                 | Meaning (positive form)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OP= <i>n</i>                         | Resets the maximum number of optimization loops that the assembler will perform if the OPNOP flag is set. The assembler will discontinue looping if there is a pass in which no optimization occurs.<br>(default: OP=3)                                                                                                                                                                                                                                                                                                            |
| OPNOP                                | Removes for optimization purposes any NOPs that have been generated by the assembler because of forward references. The assembler will often assume the worst case in determining the size of forward-referenced variables and will reserve the maximum space (usually 32 bits) for them. During the second pass of the assembler, NOPs will be generated if the operand represented by the variable can be used in an instruction format that is more compact than the size reserved during the first pass.<br>(default: NOOPNOP) |
| P= <i>type</i> [/ <i>cotype</i> ]... | Specifies a set of compatible processors and coprocessors. The <i>type</i> can be 68000, 68008, 68010, CPU32, 68020, 68030, or 68040.<br><br>The <i>cotype</i> can be 68881 (used for 68881 or 68882) or 68851 (compatible with all <i>types</i> except 68030 and 68040).<br>(default: 68000/68881)<br><br><b>Example:</b><br><br>P=68020/68881/68851<br><br>It is an error to specify an incompatible set of processors, such as P=68030/68851.                                                                                   |
| PCO<br>P                             | Uses the program counter relative addressing mode on backward references within an absolute (ORG) section whenever possible. This flag is only applied to instructions where the relative addressing mode is legal and the displacement from the program counter fits within the 16-bit field provided.<br>(default: NOPCO)                                                                                                                                                                                                        |

(cont.)

Table 2-2. VMS Assembler Command Line Flags (cont.)

| Flag     | Meaning (positive form)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PCR      | Uses program counter relative addressing mode on references from a relocatable section to the same section within the current module for all instructions for which this is a legal address mode. PCR differs from PCS in that PCS applies to all relocatable sections within this module and assumes you know the boundaries. PCR will only affect intrasection expressions within the current module (expressions for which the assembler has enough information to generate the correct relative offset).<br>(default: PCR)                                                                                                                                                                                                                                                                             |
| PCS<br>R | Uses program counter relative addressing mode on backward references from a relocatable section to the same section or to a different relocatable section. This addressing mode will be used on all instructions for which the program counter relative addressing mode is legal.<br>(default: NOPCS)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| QUICK    | Changes the MOVE, ADD, and SUB instructions to the more efficient MOVEQ, ADDQ, and SUBQ instructions.<br><br><div style="display: flex; justify-content: space-around;"> <div> <p><b>Before the Change</b></p> <p>MOVE.L #data, Dn<br/>ADD #data, ea<br/>SUB #data, ea</p> </div> <div> <p><b>After the Change</b></p> <p>MOVEQ #data, Dn<br/>ADDQ #data, ea<br/>SUBQ #data, ea</p> </div> </div> <p>where:</p> <p><i>data</i>    Legal values are -128 to 127 for MOVE and MOVEQ instructions.<br/>          Legal values are 1 to 8 for ADD, ADDQ, SUB, and SUBQ instructions.</p> <p><i>ea</i>       Effective address.</p> <p><i>n</i>        Number of data register.</p> <p>For more information on these instructions, see a <i>Motorola Microprocessor User's Manual</i>.<br/>(default: QUICK)</p> |

(cont.)

Table 2-2. VMS Assembler Command Line Flags (cont.)

| Flag  | Meaning (positive form)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REL32 | <p>Forces the assembler to default to 32-bit base and outer displacements when the address range is 32 bits long. This flag can be turned on and off at any time in the program. It will only be effective if the processor type is set to 68020 or 68030.</p> <p>Addressing modes which first appeared with the 68020:</p> <p>((bd,An,Xn) ([bd,An,Xn],od) ([bd,An],Xn,od)<br/>           (bd,PC,Xn) ([bd,PC,Xn],od) ([bd,PC],Xn,od))</p> <p>defaulted the size of the outer and base displacements to word for forward references, external references, complex expressions, and relocatable expressions. While the code produced was smaller, you had to always size cast displacement expressions if you had a processor that accessed a full 32-bit address range (68020/30). The REL32 flag lets you change the default to 32 bits without size casting displacement expressions.<br/>           (default = NOREL32)</p> |
| S     | <p>Lists the source text on the output listing.<br/>           (default = S)</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| T     | <p>Lists the symbol table on the output listing.<br/>           (default = T)</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| W     | <p>Prints warnings during the assembly.<br/>           (default = W)</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| X     | <p>Lists the cross-reference table on the output listing. This option overrides the symbol table output option, T. If both T and X are used, a cross-reference table is generated.<br/>           (default = NOX)</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |



## Invocation Examples

The examples in this section show how to invoke the assembler and specify multiple flags on the command line.

### Example:

```
$ A68K MOD68
```

The source file MOD68.SRC is assembled, producing the default output files MOD68.LIS (listing file) and MOD68.OBJ(output object module).

### Example:

```
$ A68K/FLAGS="CRE,G" MULTD
```

The source file MULTD.SRC is assembled, producing the output listing MULTD.LIS. The output object module is named MULTD.OBJ. The CRE and G options force a cross-reference table to be printed on the output listing and assembler-generated symbols to be included in the cross-reference table.

### Example:

```
$ A68K/FLAGS="-C,P=68020"/NOLIST/OBJECT=TEMP DIVD
```

The source file DIVD.SRC is assembled, producing the output object module TEMP.OBJ. Generation of an output listing is inhibited by the NOLIST option. The -C option prevents the listing after the first line of all lines of object code that are generated by the DC directive. The P=68020 option lets code be assembled for the 68020 or equivalent microprocessor.

### Example:

```
$ A68K/HP MOD1/FLAGS="P=68020,X"
```

The source file MOD1.SRC is assembled, producing the output listing MOD1.LIS. The relocatable object module produced is MOD1.OBJ.

The /HP option causes the asmb\_sym file MOD1.A to be produced as well. The assembler automatically sets the D flag, which includes local symbols in the output object module. The x flag causes the output listing to include the cross-reference table. The P=68020 flag defines assembled code for the 68020 microprocessor.

## LNK68K Linker

There are two ways to invoke the linker: using a command file and or using the command line (with an optional command file). Using these methods, you can produce absolute or relocatable output suitable for incremental linking. However, you should remember that if you use the incremental link option, which produces a relocatable object module, the only linker commands that can be entered are LOAD commands.

The next section describes the invocation syntax for the LNK68K Linker. Subsequent sections provide examples of different invocation methods.

### Invocation Syntax

Abbreviations for options are also shown. After entering the abbreviation, any number of characters can be specified up to and including the option's full spelling.

Use parentheses to specify multiple settings for the same option. For example, /COMMAND=(cmd1,cmd2) will execute two linker commands, as specified. However, if you had entered /COMMAND=cmd1 /COMMAND=cmd2, only the last parameter is used; the first command is not executed.

Options are case-insensitive and separated by a slash (/).

#### Syntax:

```

L68K  [ /ABSOLUTE[-absolute_file] | /NOABSOLUTE ]
      [ /COMMAND=command ] [ /FORMAT=format ]
      [ /HP | /NOHP ] [ /LIBRARY=library ]
      [ /MAP=map_file | /NOMAP ] [ /OBJECT=relocatable_file ]
      [ /REFERENCE=name ] [ /VERSION ]
      { /OPTION command_file | object_file [, object_file] ... }

```

#### Description:

**L68K**                      Invokes the linker.

**ABSOLUTE** [-*absolute\_file*]  
**ABS** [-*absolute\_file*]

Produces an absolute object file with the file name indicated. If not specified, the file name defaults to the same name as the command file with an .ABS extension. (default)

**NOABS**                      Inhibits generation of an absolute object module.

**COMMAND=command**

Specifies a linker command. This command behaves as though it is placed at the top of the *command file*. Multiple linker commands can be specified.

**Note**

The **/FLAGS="flag\_list"** option, which was supported in previous versions of the linker, will not be supported in future releases. For information about this option, refer to Appendix H, *Not Supported in Future Releases*.

**FORMAT=format**

Specifies the format of the file created by the linker. Legal values for *format* are the same as those for the linker **FORMAT** command and include:

|             |                                                                                                                     |
|-------------|---------------------------------------------------------------------------------------------------------------------|
| HP          | HP-OMF format                                                                                                       |
| IEEE        | IEEE-695 format (default)                                                                                           |
| INCREMENTAL | IEEE-695 format (incremental linking)                                                                               |
| S           | Motorola S-Record                                                                                                   |
| NOABS       | No object file is produced; however, internal processing is performed and a map file will be produced if requested. |

**HP**

Produces HP 64000-compatible object modules (HP-OMF) and *link\_sym* files. Sets **LISTABS PUBLICS** and **LISTABS INTERNALS**.

The default file extension for the absolute file generated is **.X**. The *link\_sym* file is given the source file root name with a **.L** extension.

For more information on this option, see the **FORMAT HP** linker command in the *Microtec Research ASM68K Reference Manual*. For more information on HP 64000 development support, see Appendix A.

**NOHP**

Does not produce an HP 64000 compatible (HP-OMF) *asmb\_sym* file. This option may be specified before or after the *source file*.

For more information on HP 64000 development support, see Appendix A.

- LIBRARY=*library*** Specifies a library to be searched after all object files have been loaded.
- MAP [-*map\_file*]**  
**M[-*map\_file*]**  
Produces a link map with the file name indicated. The default file name is the name of the source file with the extension .MAP.  
(default: *source\_file*.MAP)
- NOMAP** Does not produce an output link map.
- OBJECT [-*relocatable\_file*]**  
**OBJ [-*relocatable\_file*]**  
Produces a relocatable object file that can be linked with other object modules. The file name must be specified in batch mode. The default file name extension is .OBJ.  
  
This option can be used in place of /ABSOLUTE to incrementally link object module files. Since an absolute link is not performed, unresolved external references to other modules can exist in the resulting relocatable object module.  
  
For more information on this option, see the **FORMAT INCREMENT** linker command in the *Microtec Research ASM68K Reference Manual*.
- OPTION *command\_file***  
**OP *command\_file***  
Indicates use of a batch command file. The command file name extension defaults to .OPT.
- REFERENCE=*name*** Creates an external reference called *name* for the linker to resolve. For more information on this option, see the **EXTERN** linker command in the *Microtec Research ASM68K Reference Manual*.
- VERSION** Displays version number of LNK68K and then exits.
- object\_file*** Specifies the source object file(s). The .OBJ file name extension is assumed.

If you do not specify options, the following defaults will be used:

```
/ABSOLUTE=object_file.ABS  
/MAP=object_file.MAP  
/NOHP
```

Refer to the *Invocation Examples* section for examples of the interactive and non-interactive linker invocation modes.

It is illegal to specify certain combinations of command line options. For example, it is illegal to specify both ABS and OBJ, or OBJ and HP. The linker displays an appropriate error message if an illegal combination is specified.

## File Name Defaults

Specification of the map and output object module file names is optional. If they are not specified, the linker produces them by default. If an output file name is not specified, the linker uses either the command file name (*command\_file.extension*), if present, or the first input object file name (*object\_file.extension*) as the root file name for generated files. If file name extensions (input or output) are not specified, the default extensions listed in Table 2-3 are assumed.

Table 2-3. VMS Default Linker File Name Extensions

| File                               | Extension |
|------------------------------------|-----------|
| Absolute output object file        | .ABS      |
| Output HP-OMF link_sym file        | .L        |
| Output link map file               | .MAP      |
| Input object file                  | .OBJ      |
| Relocatable output object file     | .OBJ      |
| Input command file                 | .OPT      |
| Output HP-OMF absolute object file | .X        |

The linker uses the current defaults if you do not specify a node, device, or directory.

## Invocation Examples

The following examples illustrate various methods of invoking and using the linker. Examples shown include multiple flags entered on the command line and specifying incremental linking.

### Invoking LNK68K with a Command File

You can enter linker commands in batch mode through the use of a command file. Any fatal error in the command file prevents the link from being completed. However, all commands in the file are processed and checked for errors.

#### Example:

```
$ L68K/OPTION CMD/MAP=MY.MAP
```

The linker reads the command file `CMD.OPT` and produces a link map called `MY.MAP`. The absolute object file `CMD.ABS` is generated in *IEEE-695* format.

#### Example:

```
$ L68K/NOABS/OPTION LMULT/REFERENCE=TEST1
```

The linker reads the command file `LMULT.OPT` and produces a link map called `LMULT.MAP`. Generation of an object module is inhibited by the `/NOABS` option. It may be useful to inhibit generation of an object module in order to check for load errors.

The external symbol `TEST1` is defined which the linker tries to resolve.

#### Example:

```
$ L68K/OPTION/HP LMOD1.CMD
```

The command file `LMOD1.CMD` is read, and the link map produced is named `LMOD1.MAP`.

The `LISTABS PUBLICS` and `LISTABS INTERNALS` linker commands are automatically set by the use of the `/HP` option, so the linker includes external and local symbols in the output file. The `/HP` option generates the absolute object module, `LMOD1.X` in *HP-OMF* format, and the HP link\_sym file, `LMOD1.L`. External and local symbols are placed in the output object module.

**Example:**

```
$ L68K/HP/OPT LFILE
```

The command file `LFILE.OPT` is read, and the link map produced is named `LFILE.MAP`. If an extension had been specified with the command file name, then that command file with the specified extension would have been read. The `/HP` option causes generation of the absolute object module `LFILE.X` in HP-OMF format and the HP link\_sym file `LFILE.L`.

The `LISTABS PUBLICS` and `LISTABS INTERNALS` linker commands, which include both local and external symbols in the output object module, are automatically set by the linker.

**Invoking LNK68K Without a Command File**

You can enter the names of object modules that will be linked together on the command line.

**Example:**

```
$ L68K/OBJECT=LORP/NOMAP DORP,FORP,SORP
```

The object modules `DORP.OBJ`, `FORP.OBJ`, and `SORP.OBJ` are incrementally linked forming the relocatable object module output file `LORP.OBJ`. Output of a link map is inhibited by use of the `/NOMAP` option.

## LIB68K Librarian

The Microtec Research LIB68K Object Module Librarian builds and maintains program libraries. Libraries are collections of relocatable object modules residing in a single file.

### Invocation Syntax

The command line can be continued on the next line if a minus sign (-) is the last character on the line to be continued.

Only certain library functions can be specified on the command line: ADDMOD, DELETE, REPLACE, EXTRACT, and FULLDIR. These librarian commands can be entered in any order, but the librarian processes the commands in the order specified above. Multiple object modules and files must be enclosed in double quotes.

#### Syntax:

```
B68K  [ /OPTION command_file |
      library_file [/FULLDIR[=library_listfile]]
      [/ADDMOD="file_name"[, "file_name"...]
      [/DELETE="module_name"[, "module_name"...]
      [/EXTRACT="module_name"[, "module_name"...]
      [/REPLACE="file_name"[, "file_name"...]
      [/OUTPUT=list_file] [/VERSION]
```

#### Description:

**B68K**                      Invokes the librarian.

**OPTION *command\_file***  
**OR *command\_file***

Indicates use of a command file, which contains linker commands. The command file extension is assumed to be .OPT unless otherwise specified. In the interactive mode, the command file is the terminal (TT:).

***library\_file***              Specifies the library file to be read or written. The default file extension is .LIB.

**FULLDIR[=*library\_listfile*]**

Displays the contents of the library named on the command line (*library\_file*). If the file *library\_listfile* is not specified, the library information is written to the terminal.



**ADDMOD**="file\_name" [, "file\_name"] ...

Adds the module(s) in the named files to an existing library.

**DELETE**="module\_name" [, "module\_name"] ...

Deletes the named module(s) from the library.

**EXTRACT**="module\_name" [, "module\_name"] ...

Extracts the named module(s) from the library and places them in files that have the same module name and the extension .OBJ. The modules are not deleted from the library.

**REPLACE**="file\_name" [, "file\_name"] ...

Replaces the module(s) in the library with modules having the same name from the specified file(s).

**OUTPUT**=list\_file

Specifies the output listing file. The default file extension is .LIS.

**VERSION**

Displays the version number of LIB68K and then exits.

The output will be displayed on your terminal, unless you have specified the **OUTPUT=list\_file** option.

## File Name Defaults

A summary of librarian file name default extensions is listed in Table 2-4.

Table 2-4. VMS Default Librarian File Name Extensions

| File                      | Extension |
|---------------------------|-----------|
| Library file              | .LIB      |
| Listing file              | .LIS      |
| Relocatable object module | .OBJ      |
| Input command file        | .OPT      |

## Invocation Examples

There are three ways to invoke the LIB68K Librarian: interactive mode, command line mode, and command file mode. Examples of these methods are described below.

### Invoking LIB68K Interactively

You can enter librarian commands interactively from the terminal. A help facility is available by typing `h` or `help`. When an illegal command is entered, the librarian displays an error message and provides an opportunity to re-enter the command. In this interactive mode, most librarian command errors are not fatal. Multiple object modules or files must be enclosed within double quotes.

#### Example:

```
$ B68K
```

The librarian displays a prompt (`LIB68K>`) for interactive input.

### Invoking LIB68K from the Command Line

You can enter librarian commands on the command line. This method can only be used to modify an existing library. A new library cannot be created using this method.

Only certain library functions can be specified using this method; these are `ADDMOD`, `DELETE`, `REPLACE`, `EXTRACT`, and `FULLDIR`. These librarian commands can be entered in any order, but the librarian collects the commands and processes them in a fixed order: `ADDMOD`, `DELETE`, `REPLACE`, `EXTRACT`, and `FULLDIR`. `FULLDIR` implies `DIRECTORY`.

#### Example:

```
$ B68K EXLIB.LIB/REPLACE="SYM1.OBJ,SYM2.OBJ" -  
$ /ADDMOD="MOD1.OBJ,MOD2.OBJ,MOD3.OBJ"
```

The `REPLACE` option replaces modules `SYM1.OBJ` and `SYM2.OBJ` in the library file `EXLIB.LIB`. The `ADDMOD` option adds modules `MOD1.OBJ`, `MOD2.OBJ`, and `MOD3.OBJ` to the library `EXLIB.LIB`. The librarian will execute the `ADDMOD` command, then the `REPLACE` command. These commands can be specified on the command line in any order. The line continuation character, `(-)`, continues the command-line entry on a second line.

## Invoking LIB68K with a Command File

You can read librarian commands from a command file in batch mode. The commands are read in the exact order in which they are specified. If an error is found, commands read after the first error is found are processed, checked for errors, and executed if possible. If a library file is specified, it is not generated if an error is encountered.

### Example:

```
$ B68K /OPTION COMMAND.OPT /OUTPUT=PRINTOUT
```

The library commands are all contained in the command file `COMMAND.OPT`. The output listing will be written to the file `PRINTOUT.LIS`.

## VMS Return Codes

The assembler, linker, and librarian pass a return code to VMS upon completion of program execution. The return code indicates whether the program executed properly or not. The return codes are:

- 0   Warning.  
The program ran to completion, but there were user-created warnings.
- 1   No Error.  
The program ran to completion with no user errors.
- 2   Execution Error.  
The program ran to completion, but there were user-created assembler, linker, or librarian errors.
- 4   Fatal Error.  
The program did not run to completion due to a system problem. For example, the program was not allocated the required amount of disk space, or a file read/write error occurred.



# UNIX / DOS User's Guide 3

---

## Introduction

This chapter describes the syntax for invoking the assembler, linker, and object module librarian for use on all UNIX systems, UNIX-like systems, PC-DOS systems, and MS-DOS systems.

Command line options and their syntax are described. Command line options control the generation of the output listing and object module, and turn assembler switches on and off.

The operating system prompt used in the directions is a percent sign (%). Your system may be configured with a different prompt.

The line continuation character, the backslash (\), is used to continue the command-line entry on the next line for UNIX systems only.

## ASM68K Assembler

The Microtec Research ASM68K Assembler converts assembly language programs to relocatable object code. Object modules are produced in IEEE-695 format. After assembly, object modules are combined by the linker.

## LNK68K Linker

The Microtec Research LNK68K Linker allows multiple relocatable object modules to be combined into a single relocatable module that subsequently can be re-linked with other modules (i.e., incremental linking). The linker can also combine relocatable object modules into a single absolute object module in one of the following formats:

- Motorola S-Record format
- IEEE-695 format
- Hewlett-Packard 64000 format (HP-OMF)

The LNK68K Linker has some powerful features designed for embedded applications, which provide the capability to load your software at locations that match your unique hardware environment. You can specify:

- where to load sections
- order to load sections
- size of sections to use

If one or more of the linker input files is a library of object modules, the linker automatically loads all modules from the library that are referenced by the individually-named object modules.

## LIB68K Librarian

The Microtec Research LIB68K Object Module Librarian lets you maintain a collection of relocatable object modules that reside in one file. By using the librarian, you can format and organize library files that will subsequently be used by the linker.

## ASM68K Assembler

The Microtec Research ASM68K Assembler produces object code that will be used later by the LNK68K Linker and LIB68K Librarian.

## Environment Variables

This section describes environment variables that affect ASM68K Assembler operation. Environment variables must be set before invoking the assembler.

### TMP (DOS only)

The TMP environment variable specifies a directory path for temporary files:

```
SET TMP=my_tmpdir
```

## Invocation Syntax

This section describes the command line syntax needed to invoke the ASM68K Assembler. Abbreviations for options are shown in the *Description* section. After entering an abbreviation, any number of characters can be specified up to and including the option's full spelling.

### Syntax:

```
* asm68k [-b] [-D sym[-value]] [-F flag_list] [-h] [-H asmb_symfile]
          [-I pathname...] [-l | -L] [-o object_file] [-V] source_file
```

### Description:

|        |                                                                                                                                                                                                             |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| asm68k | The invocation name of the assembler.                                                                                                                                                                       |
| -b     | Stores intermediate information in a temporary file (instead of virtual memory) between pass 1 and 2 of the assembler. This option is valid for UNIX systems only; DOS systems always use a temporary file. |

- D *sym* [*-value*]** Defines the symbol *sym* at assembly time. The parameter *value* must be a constant. If *value* is not specified, *sym* is set to 1.
- The value of *sym* will be overridden by the first SET directive applied to *sym* in the source file. No EQU directives can be applied to *sym*.
- F *flag\_list*** Enables and disables the internal assembler control flags (see Table 3-2).
- h** Generates an HP 64000 compatible (HP-OMF) *asmb\_sym* symbol file. The *d* flag is automatically set, placing local symbols in the output object module. The default file name is *source\_file.A*.
- For more information on HP 64000 development support, see Appendix A.
- H *asmb\_symfile*** Generates an HP 64000 compatible (HP-OMF) *asmb\_sym* symbol file with the name *asmb\_symfile*. The *d* flag is automatically set, placing local symbols in the output object module.
- For more information on HP 64000 development support, see Appendix A.
- I *pathname*** Specifies a path to be searched to locate files that are to be included using the INCLUDE directive. For more information on the INCLUDE directive, see *Assembler Directives* in the *Microtec Research ASM68K Reference Manual*.
- The assembler first searches for include files in the directory containing the assembler source program. It next searches the directory specified by *pathname*. A maximum of four paths can be specified. For instance, *-ipath1 -ipath2* and so on.
- l**
- L** Produces a listing file which is written to the standard output device. The standard output can be redirected to a file.
- o *object\_file*** Overrides the default object module file name with the specified file name. The default file name is *source\_file.o* (UNIX) or *source\_file.obj* (DOS). You must use this option if you redirect the source file.

- v** Displays the version number of ASM68K then exits.
- source\_file*** Specifies the assembler source file name. The default file name extension is *.s* (UNIX) or *.src* (DOS). The source file must be the last entry on the command line, excluding redirected standard output listings.

If you do not specify options, the following defaults will be used:

*source\_file.s* (UNIX) or *source\_file.src* (DOS) or the standard input device  
 no listing file  
*source\_file.o* (UNIX) or *source\_file.obj* (DOS) in IEEE-695 format

Refer to Table 3-2 for the default flags that will be used.

Error messages as a result of erroneous command line entries are written to the standard error output device.

## File Name Defaults

If no source file is named, the source is read from the standard input device. An output object module is produced by default. The assembler uses the source file name as the root file name and appends the default file name extensions listed in Table 3-1.

Table 3-1. UNIX/DOS Default Assembler File Name Extensions

| File                    | UNIX      | DOS         |
|-------------------------|-----------|-------------|
| Assembler source file   | <i>.s</i> | <i>.src</i> |
| HP 64000 asmb_sym file  | <i>.A</i> | <i>.A</i>   |
| Relocatable object file | <i>.o</i> | <i>.obj</i> |

File names can include the following full file specification information:

Path name (current default assumed)  
 File name (current default assumed)  
 Extension (current default assumed)



## Command Line Flags

Flags specified on the command line have the effect of inserting an assembler control at the beginning of the source file.

Assembler command line flags for *flag\_list* in the `-f flag_list` option are listed in Table 3-2. Abbreviations for the flags are also shown. After entering an abbreviation, any number of characters can be specified up to the flag's full spelling. Some flags may be negated by using the prefix `no`, as shown in the table.

Multiple items in the list are separated from each other by a space, slash, or comma. If multiple flags or flags with parenthesized text are specified, the flag list should be enclosed within double quotes. Parentheses, when shown, are required. If conflicting flags are specified, the first flag overrides the second one.

More information about the flags can be obtained in the description of the `OPT` directive in the *Microtec Research ASM68K Reference Manual*. Flags entered on the command line have the effect of inserting an `OPT` directive at the beginning of the source file. However, flags on the command line do not override other `OPT` directives in the source file.

Table 3-2. UNIX/DOS Assembler Command Line Flags

| Flag     | Meaning (positive form)                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| abspcadd | When this flag is set, an absolute expression in conjunction with the mnemonic <code>PC</code> (i.e., <code>5(PC)</code> ) refers to an absolute address accessed through <code>PC</code> relative mode rather than to a relative displacement to the current program counter.<br><br>For more information, refer to <i>Assembler Syntax for Effective Address Fields</i> in Chapter 4, <i>Addressing Modes</i> , of your <i>ASM68K Reference Manual</i> .<br>(default: abspcadd) |
| brb      | Forces forward references in relative branch instructions ( <code>Bcc</code> , <code>BRA</code> , <code>BSR</code> ) to use the short form of the instruction (8-bit displacement). The <code>brb</code> flag is legal only in 68020 mode.<br>(default: nobrb)                                                                                                                                                                                                                    |

(cont.)

Table 3-2. UNIX/DOS Assembler Command Line Flags (cont.)

| Flag       | Meaning (positive form)                                                                                                                                                                                                                                                   |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| brl        | Forces forward references in relative branch instructions (Bcc, BRA, BSR) to use the long address (32 bits). If OPT OLD is in effect in conjunction with the brl flag, the longer form is used. This flag only applies to 68020 and 68030 processors.<br>(default: nobrl) |
| brs<br>b   | Forces forward references in relative branch instructions (Bcc, BRA, BSR) to use the short form of the instruction (8-bit displacement).<br>(default: nobrs)                                                                                                              |
| brw        | Forces 16-bit displacements to always be used in relative branch instructions (Bcc, BRA, BSR) that have forward references.<br>(default: brw)                                                                                                                             |
| case<br>ca | Retains case-sensitivity of symbols. For example, abc and ABC are treated as different symbols.<br>(default: case)                                                                                                                                                        |
| cex<br>c   | Lists all lines of object code that are generated by the DC directive. The nocex flag only lists the first line of a DC directive.<br>(default: cex)                                                                                                                      |
| cl<br>i    | Lists instructions that are not assembled due to conditional assembly statements.<br>(default: cl)                                                                                                                                                                        |
| cre        | Lists the cross-reference table on the output listing. This option overrides the symbol table output option, t. If both t and cre are used, a cross-reference table is generated.<br>(default: nocre)                                                                     |
| d          | Places local symbols in the output object module. This flag is automatically set if you generate an asmb_sym file by using the -H or -h option.<br>(default: nod)                                                                                                         |

(cont.)

Table 3-2. UNIX/DOS Assembler Command Line Flags (cont.)

| Flag                   | Meaning (positive form)                                                                                                                                                                |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>e</b>               | Lists lines with errors on the standard error output device, as well as on the output listing.<br>(default: e)                                                                         |
| <b>fri</b>             | Forces instructions that contain a forward reference to an absolute (non-relocatable) address to use a 32-bit address instead of a 16-bit address.<br>(default: 32-bit address or fri) |
| <b>fri</b><br><b>f</b> | Forces instructions that contain a forward reference to an absolute (non-relocatable) address to use a 16-bit address instead of a 32-bit address.<br>(default: nofri)                 |
| <b>g</b>               | Lists assembler-generated symbols in the symbol or cross-reference table. If d is also specified, these symbols are included in the object module as well.<br>(default: nog)           |
| <b>llen=<i>n</i></b>   | Sets the length of the line on the source listing to be <i>n</i> columns long. The <i>n</i> can be a number between 37 and 1120, inclusive.<br>(default: llen=132)                     |
| <b>mc</b>              | Prints macro calls on the program listing.<br>(default: mc)                                                                                                                            |
| <b>md</b>              | Prints macro definitions on the program listing.<br>(default: md)                                                                                                                      |
| <b>mex</b><br><b>m</b> | Lists macro and structured control directive expansions on the program listing.<br>(default: mex)                                                                                      |
| <b>nest=<i>n</i></b>   | Sets <i>n</i> nesting levels for macros. The maximum number of nesting levels allowed is 100 (UNIX) or 8 (DOS).<br>(default: nest=100 (UNIX) or nest =8 (DOS))                         |

(cont.)

Table 3-2. UNIX/DOS Assembler Command Line Flags (cont.)

| Flag                                     | Meaning (positive form)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>o</b>                                 | Produces the output object module.<br>(default: o)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>old</b>                               | Forces the assembler to generate 16-bit displacements for <i>Bcc</i> instructions when <i>OPT BRL</i> or <i>OPT -B</i> are specified, or when explicit <i>.L</i> qualifiers are used in 68020 mode. This is useful when migrating 68000 programs to the 68020 microprocessor.<br>(default: old)                                                                                                                                                                                                                                    |
| <b>op=<i>n</i></b>                       | Resets the maximum number of optimization loops that the assembler will perform if the <i>opnop</i> flag is set. The assembler will discontinue looping if there is a pass in which no optimization occurs.<br>(default: op=3)                                                                                                                                                                                                                                                                                                     |
| <b>opnop</b>                             | Removes for optimization purposes any NOPs that have been generated by the assembler because of forward references. The assembler will often assume the worst case in determining the size of forward-referenced variables and will reserve the maximum space (usually 32 bits) for them. During the second pass of the assembler, NOPs will be generated if the operand represented by the variable can be used in an instruction format that is more compact than the size reserved during the first pass.<br>(default: noopnop) |
| <b>p=<i>type</i>[/<i>cotype</i>]....</b> | Specifies a set of compatible processors and coprocessors. The <i>type</i> can be 68000, 68008, 68010, CPU32, 68020, 68030, or 68040.<br><br>The <i>cotype</i> can be 68881 (used for 68881 or 68882) or 68851 (compatible with all <i>types</i> except 68030 and 68040).<br>(default: 68000/68881)<br><br><b>Example:</b><br><br>p=68020/68881/68851<br><br>It is an error to specify an incompatible set of processors, such as p=68030/68851.                                                                                   |

(cont.)

Table 3-2. UNIX/DOS Assembler Command Line Flags (cont.)

| Flag     | Meaning (positive form)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pco<br>p | Uses the program counter relative addressing mode on backward references within an absolute (ORG) section whenever possible. This flag is only applied to instructions where the relative addressing mode is legal and the displacement from the program counter fits within the 16-bit field provided.<br>(default: nopco)                                                                                                                                                                                                                                            |
| pcr      | The program counter relative addressing mode will be used on references from a relocatable section to the same section within the current module for all instructions for which this is a legal addressing mode.<br><br>The pcr flag differs from pcs in that pcs applies to all relocatable sections within this module and assumes you know the boundaries. The pcr flag will only affect intrasection expressions within the current module (expressions for which the assembler has enough information to generate the correct relative offset).<br>(default: pcr) |
| pcs<br>r | Uses the program counter relative addressing mode on backward references from a relocatable section to the same section or to a different relocatable section. This addressing mode will be used for all instructions for which the program counter relative addressing mode is legal.<br>(default: nopcs)                                                                                                                                                                                                                                                             |

(cont.)

Table 3-2. UNIX/DOS Assembler Command Line Flags (cont.)

| Flag              | Meaning (positive form)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                   |                  |                  |                 |               |                |               |                |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|------------------|------------------|-----------------|---------------|----------------|---------------|----------------|
| quick             | <p>Changes the MOVE, ADD, and SUB instructions to the more efficient MOVEQ, ADDQ, and SUBQ instructions.</p> <table> <tr> <th>Before the Change</th><th>After the Change</th></tr> <tr> <td>MOVE.L #data, Dn</td><td>MOVEQ #data, Dn</td></tr> <tr> <td>ADD #data, ea</td><td>ADDQ #data, ea</td></tr> <tr> <td>SUB #data, ea</td><td>SUBQ #data, ea</td></tr> </table> <p>where:</p> <p><i>data</i> Legal values are -128 to 127 for MOVE and MOVEQ instructions.</p> <p>Legal values are 1 to 8 for ADD, ADDQ, SUB, and SUBQ instructions.</p> <p><i>ea</i> Effective address.</p> <p><i>n</i> Number of data register.</p> <p>For more information on these instructions, see a <i>Motorola Microprocessor User's Manual</i>.<br/>(default: quick)</p>                                                                                                                                              | Before the Change | After the Change | MOVE.L #data, Dn | MOVEQ #data, Dn | ADD #data, ea | ADDQ #data, ea | SUB #data, ea | SUBQ #data, ea |
| Before the Change | After the Change                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                   |                  |                  |                 |               |                |               |                |
| MOVE.L #data, Dn  | MOVEQ #data, Dn                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                   |                  |                  |                 |               |                |               |                |
| ADD #data, ea     | ADDQ #data, ea                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                   |                  |                  |                 |               |                |               |                |
| SUB #data, ea     | SUBQ #data, ea                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                   |                  |                  |                 |               |                |               |                |
| rel32             | <p>Forces the assembler to default to 32-bit base and outer displacements when the address range is 32 bits long. This flag can be turned on and off at any time in the program. It will only be effective if the processor type is set to 68020 or 68030.</p> <p>Addressing modes which first appeared with the 68020:</p> <p>((bd,An,Xn) ([bd,An,Xn],od) ([bd,An],Xn,od)<br/>(bd,PC,Xn) ([bd,PC,Xn],od) ([bd,PC],Xn,od))</p> <p>defaulted the size of the outer and base displacements to word for forward references, external references, complex expressions, and relocatable expressions. While the code produced was smaller, you had to always size cast displacement expressions if you had a processor that accessed a full 32-bit address range (68020/30). The rel32 flag lets you change the default to 32 bits without size casting displacement expressions.<br/>(default: norel32)</p> |                   |                  |                  |                 |               |                |               |                |

(cont.)

Table 3-2. UNIX/DOS Assembler Command Line Flags (cont.)

| Flag | Meaning (positive form)                                                                                                                                                                            |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| s    | Lists the source text on the output listing.<br>(default: s)                                                                                                                                       |
| t    | Lists the symbol table on the output listing.<br>(default: t)                                                                                                                                      |
| w    | Prints warnings during the assembly.<br>(default: w)                                                                                                                                               |
| x    | Lists the cross-reference table on the output listing. The x option overrides the symbol table output option, t. If both t and x are used, a cross-reference table is generated.<br>(default: nox) |

## Invocation Examples

The examples in this section show how to invoke the assembler and specify multiple flags on the command line.

File name extensions in these examples are UNIX-specific and may be different for DOS.

### Example:

```
% asm68k -h -f frl,p=68020 -l > multd.l multd.s
```

The source file `multd.s` is assembled. The `-l` option produces a listing file which is redirected from the standard output device to `multd.l`. The output object module generated by default is `multd.o`. The `-h` option causes production of an HP `asmb_sym` file named `multd.A`. The assembler will automatically set the `d` flag to place local symbols in the output object module. The `frl` flag forces instructions that contain a forward reference to an absolute address to use a 32-bit address. The `p=68020` flag identifies the target processor as 68020.

### Example:

```
% asm68k -o temp.obj -f p=68020 divd.s
```

The source file `divd.s` is assembled. The output listing is not produced because the `-l` option is not used on the command line. The output object module is named `temp.obj`. The flag `p=68020` identifies the target processor as 68020.

### Example:

```
% asm68k -h -f p=68020,x -l -o mod1.o mod1 >mod1.l
```

The source file `mod1.s` is assembled, producing the output listing `mod1.l`. The relocatable object module produced is `mod1.o`. The `-h` option causes the `asmb_sym` file `mod1.A` to be produced as well. Because the `-h` option is used, the assembler automatically places local symbols in the output object module. The `p=68020` flag specifies that assembled code is for the 68020 microprocessor. The `x` flag causes the output listing to include a cross-reference table.



## LNK68K Linker

There are two ways to invoke the linker: using a command file or using the command line (with an optional command file). Using these methods, you can produce absolute or relocatable output suitable for incremental linking. If you use the incremental link option (-r), which produces a relocatable object module, the only linker commands that can be entered are LOAD commands.

This section describes environment variables affecting the linker, the invocation syntax for the LNK68K Linker, and provides examples of different invocation methods:

### Environment Variables

This section describes environment variables that affect LNK68K Linker operation. If you use these variables, they must be set before invoking the linker.

#### MRI\_68K\_LIB

The MRI\_68K\_LIB environment variable specifies a directory path for files used by the LOAD command. The LNK68K Linker will use this path to search for libraries and object files. For more information, refer to the LOAD command in Chapter 11, *Linker Commands*, in your *Reference Manual*.

#### Note

The MRI\_68K\_LIB environment variable can only be set for one directory path at a time. If you have another application that requires the MRI\_68K\_LIB environment variable to be set to a different directory path, you will have to reset this variable before using that application.

You can set the MRI\_68K\_LIB environment variable as follows:

Using C shell (csh), enter:

```
setenv MRI_68K_LIB /my_lib_path
```

Using Bourne shell (sh), enter:

```
MRI_68K_LIB=/my_lib_path
export MRI_68K_LIB
```

For DOS, enter:

```
SET MRI_68K_LIB=my_lib_path
```

**TMP (DOS only)**

The TMP environment variable specifies a directory path for temporary files:

```
SET TMP=my_tmpdir
```

**Invocation Syntax****Syntax:**

```
lnk68k [-c command_file] [-o output_objectfile]
      [-C command] [-F format]
      [-h] [-B link_symfile] [-l library] ...
      [-m] [-M] [-r] [-u name] [-v]
      [input_object_file [, input_object_file] ...]
```

**Description:**

**lnk68k**                   The invocation name of the linker.

**-c *command\_file***

Identifies the named file as a command file, which contains one or more linker commands.

**-o *output\_objectfile***

Identifies the named file as the output object module. If this option is not specified, the default output file name is *command\_file.extension* or, if a command file is not specified, the first *input\_object\_file.extension*, where *extension* is:

.o (UNIX)

.obj (DOS)   If you use the -r option, the output object module is relocatable.

.x (UNIX)

.abs (DOS)   If -r, -B, or -h is not used, the output object module is absolute in either IEEE-695 or Motorola S-Record format.

.X

If you use the -B or -h option, the output object module is in HP format.

If -c *command\_file* is not specified, -o *output\_objectfile* must be specified to prevent overwriting the first input object file (due to the creation of the default output object module *input\_object\_file.extension*).

**-C *command*** Specifies a linker command. This command behaves as though it is placed at the top of a command file. Multiple linker commands can be specified by -C cmd1 -C cmd2, etc.

#### Note

The **-f="flag\_list"** option, which was supported in previous versions of the linker, will not be supported in future releases. For information about this option, refer to Appendix H, *Not Supported in Future Releases*.

**-F *format*** Specifies the format of the file created by the linker. Legal values for *format* are the same as those for the linker **FORMAT** command and include:

|             |                                                                                                                     |
|-------------|---------------------------------------------------------------------------------------------------------------------|
| hp          | HP-OMF format                                                                                                       |
| ieee        | IEEE-695 format (default)                                                                                           |
| incremental | IEEE-695 format (incremental linking)                                                                               |
| s           | Motorola S-Record                                                                                                   |
| noabs       | No object file is produced; however, internal processing is performed and a map file will be produced if requested. |

**-h** Produces an HP 64000-compatible absolute file (HP-OMF) and a link\_sym file. Sets LISTABS PUBLICS and LISTABS INTERNALS.

The default file extension for the absolute file generated is .X. The link\_sym file name is the source file root name with a .L extension.

For more information on this option, see the **FORMAT** HP linker command in the *Microtec Research ASM68K Reference Manual*. For more information on HP 64000 development support, see Appendix A.

- B *link\_symfile*** Produces an HP 64000-compatible absolute file (HP-OMF) and a *link\_sym* file. Sets LISTABS PUBLICS and LISTABS INTERNALS.
- The default file extension for the absolute file generated is .X. The *link\_sym* file name is the source file root name with a .L extension.
- For more information on this option, see the **FORMAT HP** linker command in the *Microtec Research ASM68K Reference Manual*. For more information on HP 64000 development support, see Appendix A.
- l *library*** Specifies a library to be searched after all object files have been loaded. Multiple libraries can be specified as follows:  
**-l file1.lib -l file2.lib.**
- m** Indicates that a map file will be written to the standard output device. The standard output can be redirected to a file.
- M** Indicates that a map file will be written to the file *command\_file.map* or *input\_object\_file.map* if there is no command file (*input\_object\_file* is the name of the first input object file specified on the command line).
- r** Generates an incremental link. If this option is not specified, an absolute file is produced. This option is equivalent to the **FORMAT INCREMENTAL** linker command.
- u *name*** Creates an external reference for the linker to resolve. This option is equivalent to the **EXTERN** linker command.
- v** Displays the version number of LNK68K and then exits.

If there is a command file and no options are specified, the following defaults will be used:

*command\_file.x* (UNIX) or *command\_file.abs* (DOS)

If there is no command file and no options are specified, the following defaults will be used:

*object\_file.x* (UNIX) or *object\_file.abs* (DOS)

Refer to the *Invocation Examples* section for examples of the interactive and non-interactive linker invocation modes.

It is illegal to specify certain combinations of command line options. For example, it is illegal to specify both **-r** and **-h**. The linker writes an appropriate error message to the standard error device if an illegal combination is specified.

## File Name Defaults

If an output file name is not specified, the linker uses either the command file name (*command\_file.extension*), if present, or the first input object file name (*input\_object\_file.extension*) as the root file name for generated files. If file name extensions (input or output) are not specified, the default extensions listed in Table 3-3 are assumed.

Table 3-3. UNIX/DOS Default Linker File Name Extensions

| File                                          | UNIX | DOS  |
|-----------------------------------------------|------|------|
| Input linker command file                     | .cmd | .cmd |
| Output HP 64000 link_sym file (-h or -H)      | .L   | .L   |
| Output map file (-M)                          | .map | .map |
| Input object file                             | .o   | .obj |
| Relocatable output object file                | .o   | .obj |
| IEEE-695 absolute output object file          | .x   | .abs |
| Motorola S-Record absolute output object file | .x   | .abs |
| HP-OMF absolute output object file (-h or -H) | .X   | .X   |

The linker uses the current defaults if you do not specify a node, device, or directory.

## Invocation Examples

The following examples illustrate various methods of invoking and using the linker. Examples shown include multiple flags entered on the command line and specifying incremental linking.

File name extensions in these examples are UNIX-specific and may be different for DOS.

### Invoking LNK68K with a Command File

This method of invoking the linker lets you specify a command file or both a command file and additional object modules on the command line. Object modules named on the command line will be loaded before any object module named in the command file is loaded. Command files let you set segment addresses, define segment loading order, and load a default set of object modules and libraries.

Command files enter linker commands in batch mode. Any error in the command file is considered fatal by the linker and the load is not completed.

**Example:**

```
% lnk68k -c lmult.cmd -m >lmult.map -F s
```

The linker reads the command file `lmult.cmd` and produces a link map `lmult.map`. The output object module produced is `lmult.x` in Motorola S-Record format.

**Example:**

```
% lnk68k -c sample.cmd -M -l lib2.lib mod1.o,mod2.o,mod3.o
```

The linker reads the command file `sample.cmd`, executes the commands in the file, and inserts three load commands which load the three object modules, `mod1.o`, `mod2.o`, and `mod3.o`. The library `lib2.lib` is then searched to resolve any remaining unresolved external references in the loaded modules. The output object module is `sample.x`, and the output map file is `sample.map`.

The file `sample.cmd` is comprised of the following code:

```
CHIP 68000
LISTMAP PUBLICS
ORDER SECT2,COMSEC
PUBLIC ARG3=$3760
* Load modules
LOAD test1.o
LOAD test2.o
LOAD lib1.lib
END
```

The command file processed by the linker will be:

```
CHIP 68000
LISTMAP PUBLICS
ORDER SECT2,COMSEC
PUBLIC ARG3=$3760
* Load modules
LOAD mod1.o
LOAD mod2.o
LOAD mod3.o
LOAD test1.o
LOAD test2.o
LOAD lib1.lib
LOAD lib2.lib
END
```

**Example:**

```
% lnk68k -c lmod1 -h -m >lmod1.m
```

The command file `lmod1` is read, and the link map `lmod1.m` is produced. If an extension had been specified with the command file name, then that command file

with the specified extension would have been read. The `-h` option generates the absolute object module `lmod1.X` in HP-OMF format and the HP link sym file `lmod1.L`. Because the `-h` option is used, the linker automatically sets the `LISTABS PUBLICS` and `LISTABS INTERNALS` linker commands, placing external and local symbols in the output object module.

**Example:**

```
% lnk68k -o sample -i -m >sample.map mod1.o mod2.o mod3.o
```

The object module files specified on the command line will be named in a `LOAD` command. This `LOAD` command will be inserted before the first `LOAD` command in the command file. If there is no command file specified, the linker constructs a command file consisting of `LOAD` commands for the object module files.

The linker loads the three object modules `mod1.o`, `mod2.o`, and `mod3.o`. The resulting incrementally-linked file is `sample.o`. The link map is `sample.map`.

The command file processed by the linker will be:

```
LOAD mod1.o
LOAD mod2.o
LOAD mod3.o
END
```

**Invoking LNK68K Without a Command File**

The object module files specified on the command line will be named in a `LOAD` command. When there is no command file specified, the linker constructs a command file consisting of `LOAD` commands for the object module files.

**Example:**

```
% lnk68k -o abc -l def.lib -i -m >abc.map mod1.o mod2.o mod3.o
```

The linker loads the three object modules `mod1.o`, `mod2.o`, and `mod3.o`. The resulting incrementally-linked file is `abc.o`. The link map is `abc.map`.

The library `def.lib` is placed at the end of all the `LOAD` commands. The command file processed by the linker will be:

```
LOAD mod1.o
LOAD mod2.o
LOAD mod3.o
LOAD def.lib
END
```

## LIB68K Librarian

The Microtec Research LIB68K Object Module Librarian builds and maintains program libraries. Libraries are collections of relocatable object modules residing in a single file.

### Invocation Syntax

The command line can be continued on the next line if a minus sign (-) is the last character on the line to be continued.

Only certain library functions can be specified on the command line: **ADDMOD**, **DELETE**, **REPLACE**, **EXTRACT**, and **FULLDIR**. These librarian commands can be entered in any order, but the librarian processes the commands in the order specified above. Multiple object modules and files must be enclosed in double quotes.

#### Syntax:

```
lib68k  [-a "filename[,filename]..."]  
        [-d "module_name[,module_name]..."]  
        [-e "module_name[,module_name]..."]  
        [-r "filename[,filename]..."]  
        [-l] [-v] library_file
```

#### Description:

|                                         |                                                                                                                                                                          |
|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>lib68k</b>                           | Invokes the librarian.                                                                                                                                                   |
| <b>-a "filename[filename]..."</b>       | Adds the module(s) in the named files to an existing library.                                                                                                            |
| <b>-d "module_name[module_name]..."</b> | Deletes the named module(s) from the library.                                                                                                                            |
| <b>-e "module_name[module_name]..."</b> | Extracts the named module(s) from the library and places them in files that have the same name as the <i>module_name</i> . The modules are not deleted from the library. |
| <b>-r "filename[filename]..."</b>       | Replaces the module(s) in the library with modules having the same name from the specified file(s).                                                                      |



- l               Lists the contents of the library named on the command line to the standard output device. The standard output can be redirected to a file.
- v               Displays the version number of LNK68K and then exits.
- library\_file*   Specifies the library to be read or written.

## File Name Defaults

The librarian supplies the default file name extensions listed in Table 3-4.

Table 3-4. UNIX/DOS Default Librarian File Name Extensions

| File                    | UNIX | DOS  |
|-------------------------|------|------|
| Command file            | .cmd | .cmd |
| Library file            | .lib | .lib |
| Listing file            | .lst | .lst |
| Relocatable object file | .o   | .obj |

This listing file contains output from the internal librarian **FULLDIR** command. This listing file should not be confused with the listing file named on the command line which does not have a default file name extension.

Default extensions are assumed when you do not append the period (.) extension. The librarian looks for a period (.) in the file name, scanning from the right to the left, and then compares the extension found to the default (lib). If they are not the same, the library will fail.

### Examples:

If the file name is `lib.lib`,

```

open lib.      (this command fails)
open lib.l     (this command fails)
open lib.lib   (this command succeeds)
open lib       (this command succeeds)

```

## Invocation Examples

There are three ways to invoke the LIB68K Librarian: command line mode, command file mode, and interactive mode. Examples of these methods are described in the sections below.

### Invoking LIB68K Interactively

You can enter librarian commands interactively from the terminal. A help facility is available by typing **h** or **help**.

#### Example:

```
% lib68k
```

The librarian prompts you for commands (**LIB68K>**). When an illegal command is entered, the librarian displays an error message and provides an opportunity to re-enter the command. In this interactive mode, most librarian command errors are not fatal. Multiple object modules or files must be enclosed within double quotes.

### Invoking LIB68K from the Command Line

You can enter librarian commands on the command line. This method can only be used to modify an existing library. A new library cannot be created using this method.

Only certain library functions can be specified on the command line: **ADDMOD**, **DELETE**, **REPLACE**, **EXTRACT**, and **FULLDIR**. **FULLDIR** implies **DIRECTORY**. These librarian commands can be entered in any order, but the librarian collects the commands and processes them in the order listed above.

#### Example:

```
% lib68k -r "sym1.o,sym2.o" -a "mod1.o,mod2.o,mod3.o" -l exlib.lib
```

The **-a** (**ADDMOD**) option adds modules **mod1.o**, **mod2.o**, and **mod3.o** to the library. The **-r** (**REPLACE**) option replaces modules **sym1.o** and **sym2.o** in the library **exlib.lib**. The contents of **exlib.lib** are listed to standard output. The **-r**, **-d**, **-a**, **-e**, and **-l** options can be entered in any order on the command line. The librarian will execute the commands in this order: **ADDMOD**, **DELETE**, **REPLACE**, **EXTRACT**, **FULLDIR**.

## Invoking LIB68K with a Command File

You can read librarian commands from a command file in batch mode. The librarian processes commands in the command file in the exact order in which they are specified.

Any error encountered when processing the command file is printed, and execution of the command that generated the error is not completed. Commands read after the first error is encountered are processed, checked for errors, and executed if possible. However, if a library file is specified, it is not generated if an error is encountered.

### Example:

```
% lib68k < command_file
```

The librarian commands in `command_file` will be read in batch mode. Any error encountered when processing the command file is printed, and execution of the command that generated the error is not completed. A command file is made up of a legal sequence of librarian commands. Refer to the *Microtec Research ASM68K Reference Manual* for more details on these commands.

## UNIX/DOS Return Codes

The assembler, linker, and librarian pass a return code to the operating system upon completion of program execution. The return code indicates whether the program executed properly or not. The return codes are:

0           Warning or No Error.

Either the program ran to completion with no user errors (No Error) or there were user-created warnings (Warning).

1           Execution Error or Fatal Error.

Either the program ran to completion with user-created assembler, linker, or librarian errors (Execution Error) or the program did not run to completion due to a system problem (Fatal Error). For example, the program was not allocated the required amount of disk space, or a file read/write error occurred.

A message describing the problem will accompany the return code for a warning, execution error, or fatal error.

## Conversion Utility

The ASM68K Assembler generates object files in the Microtec Research extended IEEE-695 format. These files must be passed through a conversion utility before they can be input to a native linker. This conversion utility converts the file into standard a.out format and is called *iee2aout*.

### Syntax:

```
iee2aout ieeefile aoutfile
```

### Description:

|                 |                                                                |
|-----------------|----------------------------------------------------------------|
| <i>iee2aout</i> | Invokes the converter.                                         |
| <i>ieefile</i>  | Specifies the file generated by ASM68K that must be converted. |
| <i>aoutfile</i> | Specifies the name of the converted object module.             |

When generating an IEEE object module to be converted by the *iee2aout* utility, the section names must be renamed using the following compiler options:

- NC (rename const variable section name)
- NI (rename initialized data section name)
- NL (rename compiler-generated literals section name)
- NS (rename string section name)
- NT (rename code section name)
- NZ (rename zerovars section name)

# HP 64000 Development System Support: Appendix A

---

## Overview

ASM68K supports generation of HP 64000 object modules and debugging files that can be downloaded and used with the HP 64000 Development System or HP 64700 emulator. The following development system instruments can be used:

- HP 64700 Emulator
- HP 64243 Emulator
- HP 64620 Logic State/Software Analyzer
- HP 64310 Software Performance Analyzer

You can generate the object module in HP-OMF format and the associated symbol files by using the `-h/ -H` options (UNIX/DOS) or the `/HP` option (VMS) on the assembler and linker command lines. For the assembler, these options cause generation of the `asmb_sym` file. When you use these options, the assembler automatically sets the debug flag, forcing local symbols to be included in the output object file. The linker automatically executes the `LISTABS PUBLICS` and `LISTABS INTERNALS` linker commands to include both external and local symbols in the output file and generates a `link_sym` file.

For more information on the HP 64000 Development System, see the *Section Types and HP 64000 Symbolic Files* section in Chapter 5, *Relocation*, of the *ASM68K Reference Manual*.

## HP 64000 Considerations

When using ASM68K for program development in conjunction with the HP 64000 development system, you should be aware of several inherent limitations of the HP 64000. These limitations are:

- The HP 64000 limits symbol names to 15 characters. Therefore, the assembler and linker truncate symbols to 15 characters and issue diagnostics for symbols whose first 15 characters are not unique.
- Symbols cannot include the characters `?`, `.`, or `$`. These characters are translated to underscores (`_`) by the assembler or are translated by the

linker in the case of library routines. This translation can result in duplicate symbol errors.

- Sections on the HP 64000 are classified as **PROG**, **DATA**, **COMN**, or **ABSOLUTE**. ASM68K sections must be mapped into these sections. ASM68K provides an option on the **SECT** assembler directive that allows the specification of the section type. These types are shown in Table A-1 along with how they are mapped into HP 64000 sections.

Table A-1. Section Types for the HP 64000 Sections

| HP 64000 Section | ASM68K Section Type |
|------------------|---------------------|
| PROG             | C (CODE)            |
| DATA             | D (DATA)            |
| COMN             | R (ROM)             |

If the section type is not explicitly defined, the assembler will attempt to make its own decision about how the section is used. It will attempt to place the code section in **PROG**, the uninitialized data section in **DATA**, and the initialized data section in **COMN**. If there are multiple code sections or multiple data sections defined in ASM68K, the second and successive sections are mapped into absolute sections. Absolute sections do not have local symbol and line number information associated with them. For this reason, when debugging, you cannot reference source lines in absolute sections.

The following considerations apply when using the HP 64000 development system (but do not apply for the 64700 emulator):

- The HP 64000 uses two special symbols in each routine (*Rlabel* and *Elabel*) as special identifiers for measurement tools. These symbols represent the return point and end address of a routine. The *label* portion of the name is the same as the routine name. These labels are generated automatically by the Microtec Research MCC68K Compiler when the /HP (VMS) or -h (UNIX/DOS) flag is specified at compile time. For programs written in assembly language, you must manually encode these labels.
- The HP 64000 microprocessor monitor program must be linked in with your code. Therefore, in order to use the 68000 family monitor program, it must be uploaded to your host computer, assembled with ASM68K, and linked with the application program by LNK68K. However, before the monitor can be assembled, it has to be modified to account for

incompatibilities between the HP assembly language and the Microtec Research/Motorola assembly language. The modifications required are minor and are summarized in Table A-2.

**Table A-2. Modifications Required for the 68000 Monitor**

| <b>Microtec Research<br/>Directive/Instruction</b> |                          | <b>Hewlett-Packard<br/>Directive/Instruction</b> |                          |
|----------------------------------------------------|--------------------------|--------------------------------------------------|--------------------------|
| TTL                                                | "68000"                  | "68000"                                          |                          |
| XDEF                                               | SYMBOL                   | GLB                                              | SYMBOL                   |
| PAGE                                               |                          | SKIP                                             |                          |
| CMP.L                                              | #MONITOR_START,2(A7)     | CMP.L                                            | #MONITOR_START,2[A7]     |
| MOVE                                               | (SP)+,PSTATUS            | MOVE                                             | [SP]+,PSTATUS            |
| DC.B                                               | 'Error! Entry Mode=User' | ASC                                              | "Error! Entry Mode=User" |
| DC.W                                               | REG_END-*                | DC.W                                             | REG_END-\$"              |
| SECTION                                            | PROG                     | PROG                                             |                          |





# Index

---

## Symbols

- (linker line continuation character) 2-20

## A

-a librarian command line option 3-20

a.out conversion 3-24

ABSOLUTE linker command line option 2-12

ABSPCADD assembler command line flag 2-5

abspcadd assembler command line flag 3-5

ADDMOD librarian command line option 2-19

### Assembler

#### UNIX/DOS

invocation examples 3-12

invocation syntax 3-2

#### VMS

invocation examples 2-11

invocation syntax 2-2

Assembler, description 1-1

## B

-b assembler command line option 3-2

BRB assembler command line flag 2-5

brb assembler command line flag 3-5

BRL assembler command line flag 2-5

brl assembler command line flag 3-6

BRS assembler command line flag 2-6

brs assembler command line flag 3-6

BRW assembler command line flag 2-6

brw assembler command line flag 3-6

## C

C linker command line option 3-15

-c linker command line option 3-14

CASE assembler command line flag 2-6

case assembler command line flag 3-6

CEX assembler command line flag 2-6

cex assembler command line flag 3-6

CL assembler command line flag 2-6

cl assembler command line flag 3-6

### Command line flags

#### assembler

UNIX/DOS 3-5

VMS 2-5

### Command line options

#### UNIX/DOS

assembler 3-2

librarian 3-20

linker 3-14

#### VMS

assembler 2-2

librarian 2-18

linker 2-12

COMMAND linker command line option 2-13

Components of the ASM68K package 1-1

### Continuation on command line

minus sign 2-18, 3-20

### Conventions, notational vi

Converting to a.out 3-24

CRE assembler command line flag 2-6

cre assembler command line flag 3-6

## D

D assembler command line flag 2-6

d assembler command line flag 3-6

-D assembler command line option 3-3

-d librarian command line option 3-20

Data flow program diagram 1-3

### Defaults, file name

(see File name defaults)

DEFINE command line option 2-2

DELETE librarian command 2-19

DELETE librarian command line option 2-19

Disabling command line flags 3-5

Documentation, description v

### DOS

(see UNIX/DOS)

**E**

E assembler command line flag 2-6  
 e assembler command line flag 3-7  
 -e librarian command line option 3-20  
 Enabling command line flags 3-5  
 Environment variables  
   MRI\_68K\_LIB 3-13  
   TMP 3-2, 3-14  
 EXTRACT librarian command 2-19  
 EXTRACT librarian command line option 2-19

**F**

-f assembler command line option 3-3  
 -F linker command line option 3-15  
 File name defaults  
   assembler  
     UNIX/DOS 3-4  
     VMS  
   librarian  
     UNIX/DOS 3-21  
     VMS 2-19  
   linker  
     UNIX/DOS 3-17  
     VMS 2-15  
 flag\_list (for assembler command line) 2-5  
 FLAGS command line option 2-3  
 FORMAT linker command line option 2-13  
 FRL assembler command line flag 2-7  
 frl assembler command line flag 3-7  
 FRS assembler command line flag 2-7  
 frs assembler command line flag 3-7  
 FULLDIR librarian command 2-18  
 FULLDIR librarian command option 2-18

**G**

G assembler command line flag 2-7  
 g assembler command line flag 3-7  
 Guide, description v

**H**

-H assembler command line option 3-3  
 -h assembler command line option 3-3  
 -H linker command line option 3-16  
 -h linker command line option 3-15  
 HP 64000  
   linker support 2-16, A-1  
   support A-1  
 HP command line option 2-3  
 HP linker command line option 2-13

**I**

I assembler command line flag 2-6  
 i assembler command line flag 3-6  
 -I assembler command line option 3-3  
 iee2out conversion program 3-24  
 Invocation examples  
   UNIX/DOS  
     assembler  
       command line 3-12  
       multiple flags 3-12  
   VMS  
     assembler 2-11  
       command line 2-11  
       multiple flags 2-11  
     linker  
       command file 2-16  
 Invocation syntax  
   assembler  
     UNIX/DOS 3-2  
     VMS 2-2  
   librarian  
     UNIX/DOS 3-22  
     VMS 2-20  
   linker  
     UNIX/DOS 3-14  
     VMS 2-16  
 IPATH command line option 2-3

## L

- L assembler command line option 3-3
- l assembler command line option 3-3
- l command line option 3-16
- l librarian command line option 3-21
- l linker command line option 3-16
- Librarian
  - VMS
    - command line options 2-18
- Librarian, description 1-2
- LIBRARY linker command line option 2-14
- Line continuation character
  - linker 2-20
- Line continuation character (UNIX) 3-1
- Linker
  - line continuation character 2-20
  - UNIX/DOS
    - invocation syntax 3-14
  - VMS
    - invocation syntax 2-12
- Linker, description 1-1
- LIST command line option 2-3
- LLEN assembler command line flag 2-7
- llen assembler command line flag 3-7

## M

- M linker command line option 3-16
- m linker command line option 3-16
- MAP linker command line option 2-14
- MC assembler command line flag 2-7
- mc assembler command line flag 3-7
- MD assembler command line flag 2-7
- md assembler command line flag 3-7
- MEX assembler command line flag 2-7
- mex assembler command line flag 3-7
- MRL\_68K\_LIB environment variable 3-13
- Multiple flags, entering
  - assembler 2-11, 3-5

## N

- NEST assembler command line flag 2-7
- nest assembler command line flag 3-7
- NOABS linker command line option 2-12
- NOHP command line option 2-3
- NOHP linker command line option 2-13
- NOLIST command line option 2-3
- NOMAP linker command line option 2-14
- NOOBJECT command line option 2-3
- Notational conventions vi

## O

- O assembler command line flag 2-7
- o assembler command line flag 3-8
- o assembler command line option 3-3
- o linker command line option 3-14
- OBJECT command line option 2-4
- OBJECT linker command line option 2-14
- OLD assembler command line flag 2-7
- old assembler command line flag 3-8
- OP assembler command line flag 2-8
- op assembler command line flag 3-8
- OPNOP assembler command line flag 2-8
- opnop assembler command line flag 3-8
- OPTION librarian command line option 2-18
- OPTION linker command line option 2-14
- OUTPUT librarian command line option 2-19

## P

- P assembler command line flag 2-8
- p assembler command line flag 3-8
- P=68000 assembler command line flag 2-8
- p=68008 assembler command line flag 3-8
- P=68008 command line flag 2-8
- P=68010 assembler command line flag 2-8
- p=68010 assembler command line flag 3-8
- p=68020 assembler command line flag 3-8
- P=68020 command line flag 2-8
- p=68030 assembler command line flag 3-8
- P=68030 command line flag 2-8
- p=68040 assembler command line flag 3-8
- P=68040 command line flag 2-8

P=CPU32 assembler command line flag 2-8  
 p=CPU32 assembler command line flag 3-8  
 P=type/68851 assembler command line flag 2-8  
 p=type/68851 assembler command line flag 3-8  
 P=type/68881 assembler command line flag 2-8  
 p=type/68881 assembler command line flag 3-8  
 PCO assembler command line flag 2-8  
 pco assembler command line flag 3-9  
 PCR assembler command line flag 2-9  
 pcr assembler command line flag 3-9  
 PCS assembler command line flag 2-9  
 pcs assembler command line flag 3-9

## Q

Questions vii  
 QUICK assembler command line flag 2-9  
 quick assembler command line flag 3-10

## R

r assembler command line flag 3-9  
 -r librarian command line option 3-20  
 -r linker command line option 3-16  
 REFERENCE linker command line option 2-14  
 REL32 assembler command flag 2-10  
 rel32 assembler command line flag 3-10  
 REPLACE librarian command line option 2-19  
 Return codes 2-21, 3-23

## S

S assembler command line flag 2-10  
 s assembler command line flag 3-11  
 Suggestions vii

## T

T assembler command line flag 2-10  
 t assembler command line flag 3-11  
 TMP environment variable 3-2, 3-14

## U

-u linker command line option 3-16

## UNIX

### HP 64000

considerations A-2  
 invocation examples 2-16  
 limitations A-1  
 line continuation character (/) 3-1  
 return codes 3-23

### UNIX/DOS 3-1

#### assembler

#### command line flags 3-5

abspcadd 3-5  
 brb 3-5  
 brl 3-6  
 brs 3-6  
 brw 3-6  
 case 3-6  
 cex 3-6  
 cl 3-6  
 cre 3-6  
 d 3-6  
 e 3-7  
 frl 3-7  
 frs 3-7  
 g 3-7  
 i 3-6  
 llen 3-7  
 mc 3-7  
 md 3-7  
 mex 3-7  
 nest 3-7  
 o 3-8  
 old 3-8  
 op 3-8  
 opnop 3-8  
 p 3-8

- p=68008 3-8
  - p=68010 3-8
  - p=68020 3-8
  - p=68030 3-8
  - p=68040 3-8
  - p=CPU32 3-8
  - p=type/68851 3-8
  - p=type/68881 3-8
  - pco 3-9
  - pcr 3-9
  - pcs 3-9
  - quick 3-10
  - r 3-9
  - rel32 3-10
  - s 3-11
  - t 3-11
  - w 3-11
  - x 3-11
  - command line options
    - b 3-2
    - D 3-3
    - f 3-3
    - H 3-3
    - h 3-3
    - I 3-3
    - L 3-3
    - l 3-3
    - o 3-3
    - V 3-4
  - command line syntax 3-2
  - file name defaults 3-4
  - invocation syntax 3-2
  - introduction 3-1
  - librarian
    - command line input 3-20
    - command line options
      - a 3-20
      - d 3-20
      - e 3-20
      - l 3-21
      - r 3-20
      - V 3-21
    - file name defaults 3-21
  - linker
    - command line options 3-14
      - C 3-15
      - c 3-14
      - F 3-15
      - H 3-16
      - h 3-15
      - l 3-16
      - M 3-16
      - m 3-16
      - o 3-14
      - r 3-16
      - u 3-16
      - V 3-16
    - command line syntax 3-13
    - file name defaults 3-17
    - illegal option combinations 3-16
    - incremental linking 3-13
    - invocation examples 3-17
    - invocation syntax 3-14
    - using command file 3-17
    - without a command file 3-19
- ## V
- V assembler command line option 3-4
  - V librarian command line option 3-21
  - V linker command line option 3-16, 3-21
- ## VAX/VMS 2-1
- assembler
    - command line flags
      - ABSPCADD 2-5
      - BRB 2-5
      - BRL 2-5
      - BRS 2-6
      - BRW 2-6
      - CASE 2-6
      - CEX 2-6
      - CL 2-6
      - CRE 2-6
      - D 2-6
      - E 2-6
      - FRL 2-7
      - FRS 2-7

- G 2-7
- I 2-6
- LLEN 2-7
- MC 2-7
- MD 2-7
- MEX 2-7
- NEST 2-7
- O 2-7
- OLD 2-7
- OP 2-8
- OPNOP 2-8
- P=68000 2-8
- P=68008 2-8
- P=68010 2-8
- P=68020 2-8
- P=68030 2-8
- P=68040 2-8
- P=CPU32 2-8
- P=type 68881 2-8
- P=type/68851 2-8
- PCO 2-8
- PCR 2-9
- PCS 2-9
- QUICK 2-9
- REL32 2-10
- S 2-10
- T 2-10
- W 2-10
- X 2-10
- command line options
  - DEFINE 2-2
  - FLAGS 2-3
  - HP 2-3
  - IPATH 2-3
  - LIST 2-3
  - NOHP 2-3
  - NOLIST 2-3
  - NOOBJECT 2-3
  - OBJECT 2-4
  - VERSION 2-4
- command line syntax 2-2
- file name defaults 2-4
- introduction 2-1
- librarian
  - command line options
    - ADDMOD 2-19
    - DELETE 2-19
    - EXTRACT 2-19
    - FULLDIR 2-18
    - OPTION 2-18
    - OUTPUT 2-19
    - REPLACE 2-19
    - VERSION 2-19
- linker
  - command line input 2-16
  - command line options
    - ABSOLUTE 2-12
    - COMMAND 2-13
    - FORMAT 2-13
    - HP 2-13
    - LIBRARY 2-14
    - MAP 2-14
    - NOABS 2-12
    - NOHP 2-13
    - NOMAP 2-14
    - OBJECT 2-14
    - OPTION 2-14
    - REFERENCE 2-14
    - VERSION 2-14
  - file name defaults 2-15
  - illegal option combinations 2-15
  - incremental linking 2-12
  - invocation examples 2-16
  - return codes 2-21
- VERSION assembler command line option 2-4
- VERSION librarian command line option 2-19
- VERSION linker command line option 2-14
- VMS
  - assembler
    - file name defaults 2-4
    - invocation examples 2-11
    - invocation syntax 2-2
  - HP 64000 support 2-16
  - librarian
    - file name defaults 2-19

**linker**

- command line options 2-12**
- invocation syntax 2-12**
- using command file 2-16**
- without a command file 2-17**

**W**

- W assembler command line flag 2-10**
- w assembler command line flag 3-11**

**X**

- X assembler command line flag 2-10**
- x assembler command line flag 3-11**
- XRAY68K Debugger, description 1-3**





## Reader's Response

Please complete and return the following Reader's Response form to help us improve our documentation. Your comments and suggestions are always appreciated.

Product Name \_\_\_\_\_  
Your Name \_\_\_\_\_  
Company \_\_\_\_\_  
Phone Number \_\_\_\_\_

Manual Date/Version \_\_\_\_\_  
Title \_\_\_\_\_  
Address \_\_\_\_\_  
\_\_\_\_\_

What is your level of programming experience?

\_\_\_\_\_ Advanced  
\_\_\_\_\_ Intermediate  
\_\_\_\_\_ Beginner

In this language?

\_\_\_\_\_ Advanced  
\_\_\_\_\_ Intermediate  
\_\_\_\_\_ Beginner

Circle the number which best expresses your rating of the information in this manual (*5 is the highest rating*):

|                 |   |   |   |   |   |                          |   |   |   |   |   |
|-----------------|---|---|---|---|---|--------------------------|---|---|---|---|---|
| Complete?       | 1 | 2 | 3 | 4 | 5 | Easy to understand?      | 1 | 2 | 3 | 4 | 5 |
| Accurate?       | 1 | 2 | 3 | 4 | 5 | Helpful examples?        | 1 | 2 | 3 | 4 | 5 |
| Well organized? | 1 | 2 | 3 | 4 | 5 | Helpful procedures?      | 1 | 2 | 3 | 4 | 5 |
| Easy to Find?   | 1 | 2 | 3 | 4 | 5 | At an appropriate level? | 1 | 2 | 3 | 4 | 5 |
| Easy to Read?   | 1 | 2 | 3 | 4 | 5 | Overall rating?          | 1 | 2 | 3 | 4 | 5 |

3.) Which programming language do you use?

4.) Does the manual provide all the necessary information? If not, what is missing?

5.) Are more examples needed? If so, where?

6.) Did you have any difficulty understanding descriptions or wording? If so, where?

7.) Which sections of this manual are the most important? The most helpful?

8.) What are the best features of this manual?

9.) What are the areas in this manual that need improvement?

10.) Did you find any errors in this manual? (Specify section and page number.)



## **Software Performance Report Instructions**

If problems or errors are encountered when using Microtec Research's software products (including documentation), a Software Performance Report (SPR) should be completed and returned to:

**Microtec Research  
Software Performance Reports  
2350 Mission College Boulevard  
Santa Clara, CA 95054**

Any comments or suggestions you may have should also be included. All SPRs are directed to our Technical Support Department for analysis and response.

When completing the top section of the SPR, enter the following:

1. Product name and Version number as it appears on the media label or the banner that appears on the screen when the product is invoked.
2. Serial Number as it appears on the media label.
3. Name of the person (Customer) that Microtec Research can contact.
4. Name of the company that the software was Purchased From.
5. Company Name and a shipping Address.
6. Phone Number and/or FAX Number of contact name.
7. Host (machine where the Microtec Research product is run).
8. Host Operating System and Version Number.
9. Report Type showing the type of problem reported.
10. Customer Impact indicating the severity of the problem.

The following guidelines should be followed when completing the bottom section of the SPR:

1. Give as complete a description as possible of the problem, error, or suggestion, including the exact wording of any error messages.
2. If possible, isolate the problem by providing a small example.
3. If the error example is longer than one page of source code, send all information on a tape or floppy disk.
4. Carefully label any media you send and indicate whether you want it returned to you.
5. Include command files, listings, load maps, include files, data files, and all other relevant material with the SPR.
6. If applicable, send sample input and output listings.
7. If the output is from a compiler, send a mixed (source/assembly code) listing.

If the information on the SPR is complete and concise, it will help us give you accurate and timely service to any software problem.



## Software Performance Report

|                 |                      |
|-----------------|----------------------|
| Product:        | Customer:            |
|                 | Company Name/Address |
| Version:        |                      |
| Serial Number:  |                      |
| Purchased From: | Phone Number         |

Host:      ☐ Apollo      ☐ HP      ☐ PC      ☐ Other \_\_\_\_\_  
         ☐ Sun-3      ☐ Sun-4/SPARC      ☐ VMS

Operating System \_\_\_\_\_ Version \_\_\_\_\_

|                                          |                                   |
|------------------------------------------|-----------------------------------|
| Report Type:                             | Customer Impact:                  |
| <input type="checkbox"/> Problem/Error   | <input type="checkbox"/> Heavy    |
| <input type="checkbox"/> Documentation   | <input type="checkbox"/> Moderate |
| <input type="checkbox"/> Feature Request | <input type="checkbox"/> Minor    |
| <input type="checkbox"/> Other _____     |                                   |

Problem Description (Attach additional pages if needed):

Problem Recreation (Commands or lines of code used to duplicate problem):

Workaround (Attach additional pages if needed):



C

C

C



***Microtec  
Research Inc.***

2350 Mission College Blvd.

Santa Clara, CA 95054

Tel. 408.980.1300

Toll Free 800.950.5554

FAX 408.982.8266